ORNL/SPR-2024/3303

Procedure for the Computational Models Created for the DOE/NRC Collaboration for Criticality Safety Support for Commercial-Scale HALEU Fuel Cycles and Transportation (DNCSH) Project

T. M. Greene
W. A. Wieselquist

May 2024

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

Author:

____ ______________________

Travis M. Greene, Technical Professional Assc. Staff Member **Date**
Nuclear Criticality Safety

Reviewed by:

____ ______________________

LaTravia Harmon, FFESD Quality Manager **Date**
Quality Assurance Division

Approved by:

____ ______________________

Name, Director **Date**
Nuclear Energy and Fuel Cycle Division

REVISION HISTORY

Revision Number	Effective Date	Revision Description
0	Date issued in EDRM (Enterprise Document and Records Management system)	Initial version

ACRONYM LIST

DNCSH	DOE/NRC collaboration for criticality safety support for commercial-scale HALEU fuel cycle and transportation
DOE	US Department of Energy
ENDF	Evaluated Nuclear Data File
HALEU	high-assay, low-enriched uranium
ICSBEP	International Criticality Safety Benchmark Evaluation Project
JEFF	Joint Evaluated Fission and Fusion
MCNP	Monte Carlo N-Particle®
NCERC	National Criticality Experiments Research Center
NCSP	Nuclear Criticality Safety Program
NRC	US Nuclear Regulatory Commission
ORNL	Oak Ridge National Laboratory
PM	project manager
QA	quality assurance
QAC	quality assurance coordinator
SCALE	Standard Computer Analyses for Licensing Evaluation
SDF	sensitivity data file
VALID	Verified, Archived Library of Input and Data

TABLE OF CONTENTS

ABSTRACT

This procedure outlines the application models developed as part of the US Department of Energy (DOE) and Nuclear Regulatory Commission (NRC) collaboration for criticality safety support for commercial-scale HALEU fuel cycle and transportation (DNCSH) project, an effort authorized by the US Congress. The project is a joint effort among the DOE, NRC, numerous national laboratories, and private enterprises with project management from Oak Ridge National Laboratory.

1.0 INTRODUCTION

This procedure provides a framework for preparing, reviewing, and storing model inputs and derived data in a central repository so that researchers and analysts can utilize the inputs and data with confidence in their analyses. In order for the DNCSH project to prioritize the need for benchmarks for specific systems and rank potential benchmarks for funding, application models must first exist that highlight, from a criticality safety perspective, the relevant commercial-scale needs. For example, this may be an application model for transporting a large number of pebbles via tractor-trailer.

The QA procedure documented here for application models uses documented checks and reviews to ensure that the inputs and data were correctly generated using appropriate references. Configuration management is implemented to prevent inadvertent modification of the inputs and data or inclusion of models that have not been reviewed. The procedure also provides the process to be followed if errors are identified or if input or data revisions are needed.

Note that the DNCSH project is concerned with computational models from two different but related tracks: (1) models of critical benchmark experiments and (2) models of HALEU-based fuel cycle applications. This QA procedure is concerned only with (2) application models. The standard industry practice for (1) comes from the International Criticality Safety Benchmark Evaluation Project (ICSBEP) [1], an international collaboration of the Organisation for Economic Co-operation and Development member countries for maintaining benchmark experiments used by designers, regulators, standards bodies, and so forth.

Currently, and over the past decades, the database of ICSBEP critical benchmark experiments and their associated data has been used in many regulatory functions, including confirmatory evaluations for validation purposes and industrial design settings. This includes the use of various nuclear data—such as Evaluated Nuclear Data File (ENDF/B) [2] and Joint Evaluated Fission and Fusion nuclear data library (JEFF) [3]—and criticality methods from various computational methods: Standard Computer Analyses for Licensing Evaluation (SCALE) [4], Monte Carlo N-Particle (MCNP) [5], SERPENT [6], and so forth. The current ICSBEP database contains thousands of benchmark experiments from various nuclear facilities around the world utilizing a variety of fissile materials, compositions, and energy ranges. For a critical experiment to gain acceptance into the *International Handbook of Evaluated Criticality Safety Benchmark Experiments* (ICSBEP Handbook) [1], experimenters must submit their benchmarks to a rigorous standardized review process in which multiple independent reviews occur before an experiment is accepted. This ensures that each benchmark is of the highest possible quality before it is publicly released.

From a user perspective, the data supplied by the ICSBEP Handbook can be used by potential applicants to demonstrate that the computer code and performance predictors for a particular license or application are validated to specific performance metrics, including any associated quality assurance (QA) approach or regulations that must be followed. In general, data from a critical benchmark that is approved or included in the ICSBEP Handbook are considered acceptable for use; however, the user of the data is responsible for demonstrating that the specific data fit under the umbrella of their QA program and are sufficient to support the use of a selected computer code and predictive performance requirements.

2.0 PURPOSE

To design experiments that improve the validation basis, models of representative applications must be developed. The purpose of this procedure is to prescribe the process used to create and maintain those application model inputs and associated derived data at https://code.ornl.gov/dncsh/applications, which uses the ORNL GitLab issue-tracking system. Although GitLab is most often used to manage source code, it is also useful for managing any files for which version control and accompanying documentation and review process management are important.

The development of the models submitted to the repository is based on the needs or gaps of the project as identified in the validation of HALEU-based processes. This need for additional information will lead to the design and execution of experiments to address the lack of knowledge. **Figure 1** provides a flowchart for the development and execution of the DNCSH application model repository. The highlighted box is of concern with this QA procedure.

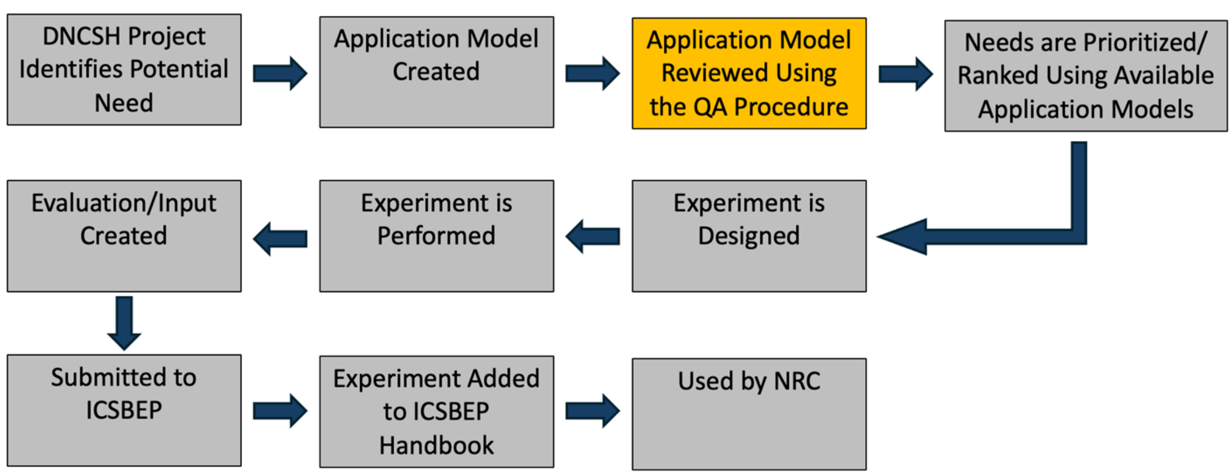

Figure 1: Flowchart for DNCSH application model inputs.

Although all national laboratories have processes that ensure all work is performed according to DOE Order 414.1D, *Quality Assurance* [8], the DOE and NRC have requested the application models developed to have an explicit, additional QA guidance document. The DNCSH application model repository will be managed according to this procedure to provide rigorous controls over all submitted material so that there is high confidence that the collection, review, and prioritization of benchmark proposals performed as a result of this project are done in a consistent manner.

The following list describes the future events that are expected to take place and for which the DNCSH project is preparing.

1. The NRC receives an application for a process or system with a criticality safety component related to 10 CFR Part 70 or 71 [9].
2. The NRC performs a confirmatory analysis, which typically involves assessing the validation basis; typically, the applicant will have performed a study with a code and nuclear dataset for a set of validation cases that are believed to be applicable.
3. The NRC and supporting institutions, such as ORNL, utilize various techniques to assess the applicability and validation basis independently.
4. The associated analyses rely heavily on critical benchmark experiments found in the ICSBEP Handbook or other high-quality sources.
5. These benchmark experiments are based on controlled critical experiments that have been carefully designed and evaluated.
6. The controlled critical experiments have been performed at various institutions and facilities around the world for many years, including current US facilities (the National Criticality Experiments Research Center at the Nevada National Security Site, the Sandia Pulsed Reactor Facility, and Critical Experiments at Sandia National Laboratories).

In order for the controlled critical experiment (6) to exist, a need must have been identified, an experiment designed and carried out, a technical review performed, and a new edition of the ICSBEP handbook created. This process has historically taken 5 to 10 years, from the time when a need was identified to when the corresponding critical benchmark is available in the ICSBEP. The US Nuclear Criticality Safety Program (NCSP) has provided funding and coordination for a majority of the critical benchmark experiments. The NCSP has a multistage process for proposing and designing experiments that meets stringent specifications. One of the steps in the experimental process is showing a need or a validation gap, such as a lack of experiments for validating graphite or chlorine in an application system.

In order to best prioritize, and hopefully reduce the cost and time, a specific need may be shown by creating a computational model for a system for which nuclear criticality safety is relevant. This model is then compared with the existing ICSBEP benchmarks to understand whether the relevant physics of the system are currently captured in the current ICSBEP handbook.

3.0 SCOPE

This procedure provides the process for preparing, reviewing, mitigating, and documenting the quality of model inputs and data derived from input models. It is specific to the DNCSH project and the ORNL GitLab system. Quality is pursued mainly through independent review by qualified experts in criticality safety modeling with SCALE. SCALE is used as the main modeling tool in this work for the following reasons:

- SCALE is the main code used by the NRC for criticality safety confirmatory analysis.
- The SCALE team already has extensive application models developed in SCALE for advanced reactor systems as part of the Volume 3 and 5 projects with the NRC.
- The SCALE team has extensive experience with HALEU applications in front-end and back-end contexts.
- SCALE has an extensive validation basis, with current work extending validation to HALEU-based fuel cycles.

Because of the short timeline of this project, focusing on consistent modeling within one application code is more efficient than using multiple codes. However, several activities will be pursued by the project to facilitate usage of MCNP, especially for experiment design.

Sensitivity coefficients and sensitivity-based similarity are used to create benchmark experiments that closely resemble the actual applications. In SCALE, this information is stored in sensitivity data files (SDFs). This project will pursue standards and conversion tools to allow sensitivity data to be shared between the two codes.

Additionally, the project will develop a SCALE-to-MCNP model conversion process, focusing on criticality safety models of interest to this project. In this way, the central SCALE models in the DNCSH repository that have gone through the QA process can be translated to MCNP for use in other systems that may help identify appropriate benchmarks (e.g., Whisper).

The following computational model QA procedure does not affect the following:

- Any aspect of or person performing the experiments, safety aspects, etc.
- The review process for potential ICSBEP benchmark experiments
- The SCALE QA plan
- The SCALE criticality safety validation case QA, VALID (Verified, Archived Library of Inputs and Data) [10]
- The NRC application review process for criticality safety

4.0 DEFINITIONS

4.1 MODEL

In this procedure, a model is the set of computational input data that is used to describe a system of interest. For example, models of critical experiments are frequently used to validate criticality safety analyses. Models may be developed for hypothetical or representative systems (e.g., the GBC-32 cask or PBMR-400 reactor) or for actual process-facility applications. A model is the complete set of inputs (geometry, materials, cross-section library, cross-section processing treatments, etc.) that represent the benchmark, advanced reactor system, or process application.

4.2 DERIVED DATA

Derived data are data produced from computer calculations using a model. Nuclide-dependent, reaction-dependent, and energy-dependent k_{eff} SDFs are examples of data derived from criticality safety analysis or advanced reactor models.

4.3 REPOSITORY

The repository is the collection of verified models and derived data that is available for use. **Figure 2** is an example from the OpenARMs repository that hosts the SCALE models developed for the NRC project assessing code readiness for advanced (non–light water) reactors [11]; this OpenARMs repository will be a basis for the DNCSH repository.

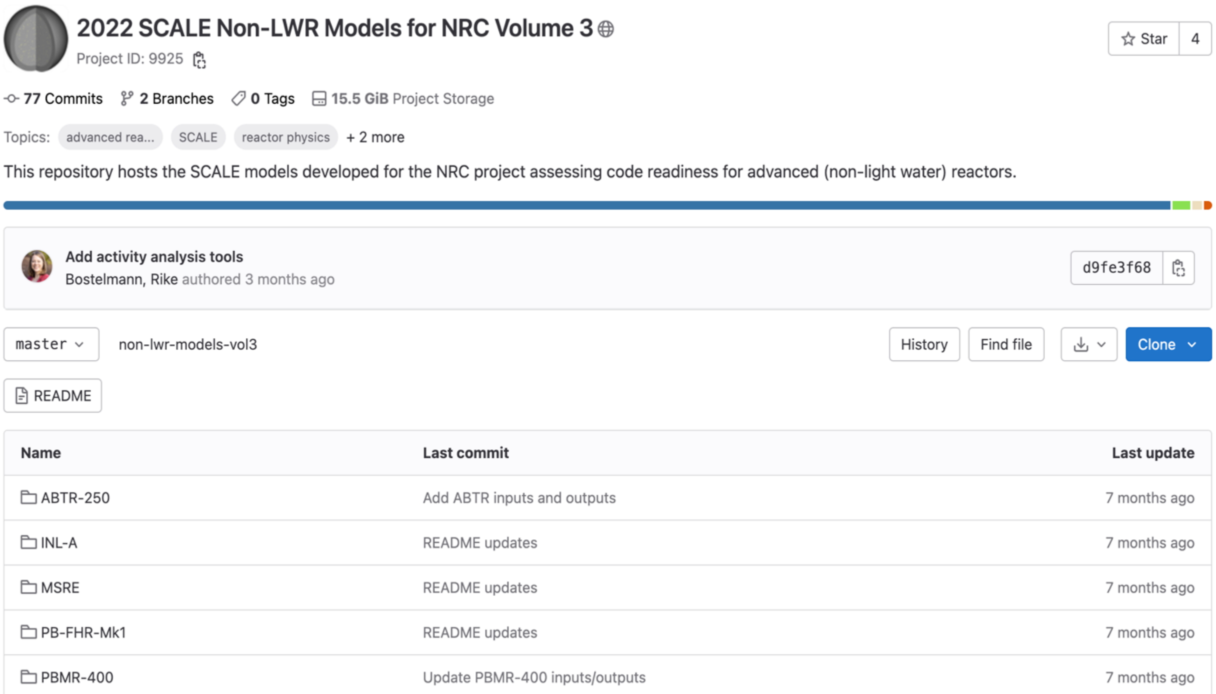

Figure 2: OpenARMs repository used as a basis for the DNCSH repository.

4.4 CASE

A DNCSH case is a set of models that moves through the procedure as a single unit. Thus, multiple models can be submitted to the repository during the origination and review of a single case. Each DNCSH case is documented in a single GitLab issue.

5.0 RESPONSIBILITIES

5.1 ORIGINATOR

The originator is responsible for providing a public reference document which includes the following.

- Methods and data used to prepare the models and derived data
- Any approximations or assumptions used in the development of models, including potential inconsistencies or inadequacies

It is expected that most of these reference documents will be ORNL TMs. It is allowed to have a document in a draft state in order that the originator can ensure the document has the detail necessary to act as a reference for this purpose.

The originator is then responsible for the following repository actions.

- Committing files to be added to the repository to the appropriate repository branches (locations) and creating merge requests for review

- Notifying the DNCSH quality assurance coordinator (QAC) of any errors or deficiencies in the references used to create the models
- Checking the models and derived data to ensure the model inputs are correct and the derived data are correctly calculated
- Submitting models and data for addition to the repository
- Resolving review comments

5.2 REVIEWER

The reviewer is responsible for the following:

- Ensuring public documentation prepared by the originator is complete and accurate
- Ensuring model references are appropriate
- Reviewing models and derived data
- Working with the originator to resolve review comments
- Documenting reviews

5.3 DNCSH QAC

The QAC is responsible for managing all aspects of the DNCSH repository, including performing the following:

- Ensuring the originator has the necessary background and experience level to create high-quality SCALE models
- Ensuring, with discussion with the DNCSH project leader, that submitted models are relevant to the project
- Ensuring the reviewer has the necessary background and experience level to review specific submitted models
- Assisting with the resolution of any unresolved review comments
- Ensuring documentation is complete
- Accepting or rejecting models and derived data
- Approving merge requests to add new files to the repository
- Reviewing problems reported with models or data in the repository
- Ensuring users are aware of their responsibilities

5.4 DNCSH NATIONAL TECHNICAL DIRECTOR

The DNCSH National Technical Director is responsible for the following:

- Assigning a qualified member as the QAC
- Providing NRC/DOE management direction in response to disagreements concerning the provisions of this procedure (e.g., required level of checking and reviewing or reference acceptability)

5.5 USERS

Although the DNCSH repository is public and anyone can view files, certain aspects of GitLab, such as notification when an issue is updated, require a registered user. By contacting the QAC, anyone can request to be registered as a user. A user in this context is given Reporter access to the repository and is responsible for the following:

- Notifying the QAC if problems are identified with models or derived data already in the repository
- Ensuring that their use of the models or derived data obtained from the repository meets the QA requirements for their specific work/analysis
- Subscribing to GitLab notifications so that questions and comments can be resolved in a timely manner

6.0 QUALIFICATION

As stated in Section 5.3, the QAC is responsible for ensuring the qualification of individuals acting as originators or reviewers for each DNCSH case. General guidance for minimum requirements for qualification is provided in this section. The QAC approves each qualification, so deviations from these recommendations are allowed on a case-by-case basis. Before performing work on a DNCSH case, however, participants should review and understand the procedural requirements in Section 7.0.

6.1 ORIGINATOR

An originator will be an experienced SCALE user for criticality safety applications. The individual will also have reviewed this procedure and provided documentation of their review and understanding of the GitLab issue for which they are the originator.

6.2 REVIEWER

A reviewer will be an experienced SCALE user for criticality safety applications and will understand the methods, strengths, and potential weaknesses of the techniques involved. The reviewer will also have reviewed this procedure and provided documentation of their review and understanding to the QAC on the GitLab issue for which they are the reviewer.

6.3 EXPIRATION OF QUALIFICATIONS

The QAC should review the list of qualified performers periodically to ensure that personnel associated with the project who should be removed from the qualified list are appropriately identified and removed. There are neither specific requirements to maintain qualification nor events that necessitate termination of qualification. There is no set time limit for qualification expiration; each performer is assigned by the QAC for each case, so qualification for each case is assessed on a case-by-case basis.

7.0 PROCEDURE

7.1 INITIAL ADDITION OF INPUTS AND DATA TO THE REPOSITORY

Step 0 – Proposed

7.1.1 The QAC is contacted by a potential originator with models and derived data to be added to the repository. The QAC assesses the qualifications of the originator and, if the originator is approved, allows the originator full write access to the DNCSH repository.

7.1.2 The originator opens a new issue in the DNCSH project within GitLab to discuss the model further, including sharing the reference document that describes the model in detail. The originator provides a brief description of the case(s) in the **Description** section of the issue template, as seen in **Figure 3**.

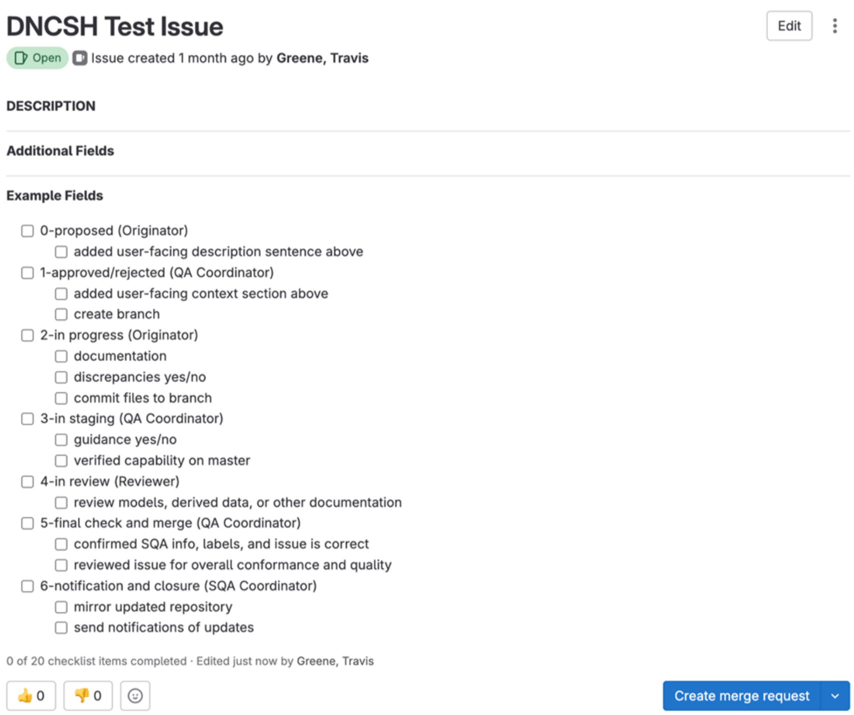

Figure 3: Description section of the case issue.

Step 1 – Approved/Rejected

7.1.3 With more information on the model, and potentially after consulting others and/or the DNCSH lead, the QAC approves or rejects the request. If the issue is approved, then the QAC adds the originator in the **Activity** section as seen in **Figure 4**. Any qualified individual may be the originator.

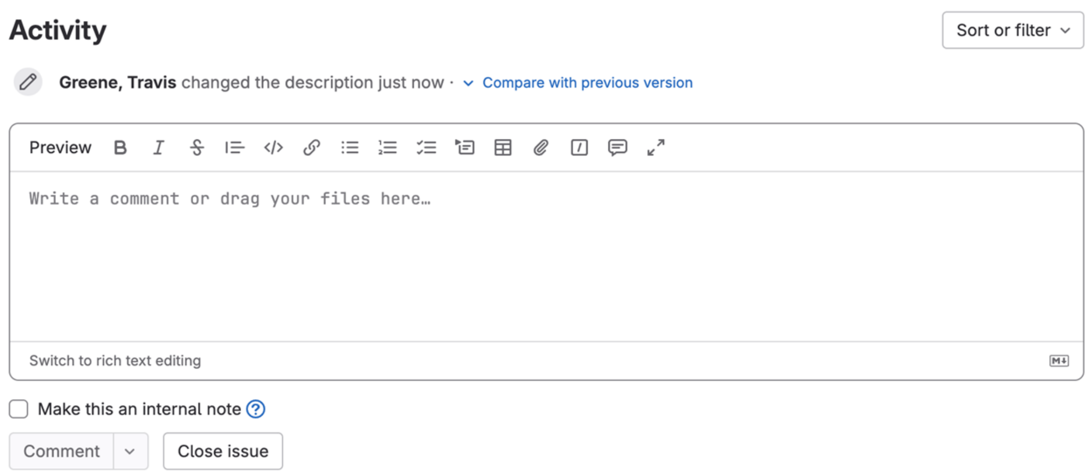

Figure 4: Activity section for making comments.

7.1.4 The QAC creates a branch *XXX-revNN* from main; in this case, *XXX* is the DNCSH applications model GitLab issue number, and *NN* is the revision number. (Revision numbers start at 01.)

7.1.5 The QAC may then assign the issue to an originator, and the label "in progress" may be added to the case. If no originator is available, then the label "approved" is added to the case instead.

7.1.6 If the issue is rejected, then the QAC notifies the originator that the issue has been rejected, documents the notification in the **Activity** section, and closes the issue. Example reasons for rejection include the following:

- The requested models or data are not appropriate for repository inclusion.
- Supporting references are not adequate.
- Work cannot commence because of funding or work priority issues.

Step 2 – In Progress

7.1.7 The originator prepares and checks the necessary information for the issue.

7.1.8 The originator completes documentation of the model(s) and derived data by using the web page markdown table and/or attaching documents directly to the issues and includes a notification in the **Activity** section. The originator includes a summary of the model and a brief description of key parameters in the GitLab issue in the appropriate sections. The originator also documents any revision number or other unique identifier(s) associated with the reference(s) used in the model creation.

7.1.9 If the originator notes potential discrepancies in the source reference descriptions, then the originator edits the field to **yes** under the **Additional Fields** section from **Figure 3**. The originator documents the issue(s) in the **Activity** section.

7.1.10 The QAC notifies the appropriate personnel responsible for further action and documents the notification in the **Activity** section.

7.1.11 The originator commits the files to the branch, opens a merge request, and edits the label with the corresponding merge request number (![Merge request number]). This indicates in the issue that origination is complete. The originator assigns the GitLab issue to the QAC.

7.1.12 The originator removes the "in progress" label and adds the "in staging" label in GitLab.

Step 3 – In Staging

7.1.13 The QAC determines the necessary level of review using a graded approach. In determining the level of review, the QAC may consider the model source, the quality of the model reference, the rigor of other reviews already performed, and the expected use of the models and data. If guidance is needed regarding the type of review required, then the QAC provides the necessary information in the **Activity** section and edits the field to **yes** in the **Additional Fields** section. If no guidance is needed regarding the type of review, then the QAC edits the field to **no**.

7.1.14 The QAC assigns the issue to the reviewer, removing the "in staging" label and adding the "in review" label. If no reviewer is available, then the issue retains the "in staging" label. Only one reviewer is needed for each set of models.

Step 4 – In Review

7.1.15 One independent reviewer reviews the models, derived data, and other documentation and provides comments in the **Activity** section (either in the **Issue** or **Merge Request [commit message – addressing feedback]** section).

7.1.16 The originator performs the necessary rework and/or addresses the reviewer's comments and provides comments in the **Activity** section (**Issue** or **Merge Request**, whichever is consistent with the option selected by the reviewer). The reviewer indicates in the **Activity** section (**Issue** or **Merge Request**) that the comments have been addressed.

7.1.17 If there is difficulty resolving comments, then the originator notifies the QAC. If significant work is needed to resolve the comments, then the QAC assigns the case to the originator and the case returns to Step 2 – In Progress. The QAC documents these actions in the **Activity** section. Moderate amounts of work can be performed to address the reviewer's comments without impacting the GitLab issue status.

7.1.18 The reviewer indicates that all comments have been resolved and the review is complete in the **Activity** section and assigns the case to the QAC. The QAC removes the "in review," and the "final check" label is added.

Step 5 – Final Check and Merge

7.1.19 The QAC reviews the issue to ensure that all necessary actions have been performed and documented. If no deficiencies are identified, then the QAC documents that the issue is complete and approved in the **Activity** section. The QAC removes the “final check” label and adds the “notification” label.

7.1.20 The QAC merges the branch onto the master branch in the repository.

Step 6 – Notification and Closure

7.1.21 The QAC sends the appropriate notifications to the DNCSH users and documents the action in the **Activity** section.

7.1.22 The QAC closes the issue.

NOTE: For additions of revised models or data to the DNCSH repository (as addressed by Section 7.4 of this procedure), the configuration control list update and the notification email will identify that the models or data are replacements for files previously removed from the repository. The configuration control list will identify the revision numbers for the files.

7.2 USE OF INPUTS AND DATA

7.2.1 Users of inputs and derived data obtained from the DNCSH repository ensure that QA requirements are satisfied for the analyses for which they use the data.

7.2.2 Users report any errors or deficiencies as described in 7.4.1.

7.3 CONFIGURATION MANAGEMENT OF INPUTS AND DATA

7.3.1 The QAC ensures that the models are compatible with new versions of SCALE and new versions of data, preferably by checking that models run with new beta releases and are consistent with previous results.

7.3.2 The QAC ensures that the models are backed up on a regular basis.

7.4 REVISIONS OR CORRECTIONS OF REPOSITORY CONTENT

Step 0 – Identification and Notification

7.4.1 The QAC is notified when a potential error in the inputs or derived data is identified in the DNCSH repository. The QAC opens a new error issue in the DNCSH project within GitLab. If an internal member of the DNCSH repository (someone with the requestor or reviewer qualifications) finds a potential error, then they may create the issue and notify the QAC of the issue itself. The error issue provides a brief description of the case(s) and reports the specific error(s) in the **Description** section of the issue template (**Figure 3**).

7.4.2 The QAC adds a “potential ERROR” label to the original issue and adds a link to the potential error issue in the original issue commentary.

NOTE: GitLab supports a subscribe option for each issue. This is the recommended way for users to receive notifications when any changes are registered in a case, they have interest in.

7.4.3 In addition, if the QAC believes the error warrants a larger distribution, then the QAC may send out a notification via email to any recipients.

Step 1 – In Review

7.4.4 The QAC assigns someone with reviewer qualifications to review the potential error or proposed revision and recommends either “no revision needed,” “revision recommended,” or “removal of inputs from repository recommended” by editing the level-1 heading and documenting the supporting reasons in the **Activity** section.

7.4.5 If no revision is needed, then the QAC removes the “potential ERROR” label. If an email was sent in the notification stage, then the QAC sends a follow-up email that the issue has been resolved, documents the notification in the **Activity** section, and closes the issue.

Step 2 – Revision

7.4.6 If a revision is recommended, then the QAC reopens the original case issue and changes the “potential ERROR” label to a “MINOR ERROR” or “MAJOR ERROR” label at their discretion. The QAC opens a new branch with name *XXX-revNNN,* where *XXX* is the original issue number and *NNN* is the new revesion number (e.g., 02 if this is the first revision) for issue *XXX* and proceeds as described in Section 7.1 with a new merge request and review. The QAC changes the label of the issue reporting from “defect” to “in testing” and documents the revision action in the **Activity** section.

Step 3 – Finalization/Removal

7.4.7 If the inputs should be permanently removed from the repository, then the QAC creates a merge request that removes the files instead of modifying them but otherwise proceeds through the same review steps. In the case of removal, it is recommended that the QAC send a notification email instead of relying solely on the GitLab notification system.

8.0 REFERENCES

[1] *International Handbook of Evaluated Criticality Safety Benchmark Experiments*, NEA/NSC/DOC (95)03, Organization for Economic Co-operation and Development, Nuclear Energy Agency, Paris, France (2022).

[2] Brown, D. A., et al., “ENDF/B-VIII.0: the 8th major release of the nuclear reaction data library with CIELO-project cross sections, new standards and thermal scattering data,” *Nuclear Data Sheets*, 148, 1-142 (2018).

[3] Plompen, A. J. M., et al., "The joint evaluated fission and fusion nuclear data library, JEFF-3.3," *European Physical Journal,* A56, 181 (2020).

[4] Wieselquist, W. A., and Lefebrve, R. A., *SCALE 6.3.0 User Manual*, Oak Ridge National Laboratory, Oak Ridge, Tennessee, ORNL/TM-SCALE-6.3.0 (2021).

[5] Kulesza, J. A., et al., *MCNP® Code Version 6.3.0 Theory & User Manual*, Los Alamos National Laboratory, LA-UR-22-3006, Rev. 1 (2022).

[6] Leppänen, J., et al., "The Serpent Monte Carlo code: Status, development and applications in 2013," *Annals of Nuclear Energy*, 82, 142-150 (2015).

[7] *Quality Assurance Requirements for Nuclear Facility Applications*, ASME NQA-1-2022, The American Society of Mechanical Engineers, New York (2022).

[8] *Quality Assurance,* DOE Order 414.1D, US Department of Energy, Washington, DC (2011).

[9] "Nuclear Regulatory Commission regulations Title 10, Code of Federal Regulations," United States Nuclear Regulatory Commission, https://www.nrc.gov/reading-rm/doc-collections/cfr/index.html. Accessed 6 March 2024.

[10] Marshall, W. J., and Reed, D. A., *SCALE Procedure for Verified, Archived Library of Input and Data (VALID)*, Oak Ridge National Laboratory, Oak Ridge, Tennessee, SCALE-CMP-012, Rev. 2 (2013).

[11] Bostlemann, F., "2022 SCALE non-LWR models for NRC Volume 3 GitLab," GitLab, code.ornl.gov/scale/analysis/non-lwr-models-vol3. Accessed 31 Oct. 2023.