

LA-UR-26-27207

Approved for public release; distribution is unlimited.

Title: MCNP6 Code Development and Software Quality Assurance

Author(s): Rising, Michael Evan

Intended for: Meeting: EFCOG Nuclear and Facility Safety Workshop, 2026-08-15 - 2026-08-21 (Oak Ridge, Tennessee, UNITED STATES)

Issued: 2026-08-11



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

MCNP6 Code Development and Software Quality Assurance

Michael E. Rising, XCP-3, LANL

2026 Nuclear Criticality Safety Community of Practice Meeting

August 17, 2026

LA-UR-26-XXXXX

MCNP[®] Trademark

MCNP[®] and Monte Carlo N-Particle[®] are registered trademarks owned by Triad National Security, LLC, manager and operator of Los Alamos National Laboratory. Any third party use of such registered marks should be properly attributed to Triad National Security, LLC, including the use of the [®] designation as appropriate.

- ▶ Please note that trademarks are adjectives and should not be pluralized or used as a noun or a verb in any context for any reason.
- ▶ Any questions regarding licensing, proper use, and/or proper attribution of Triad National Security, LLC marks should be directed to trademarks@lanl.gov.

Outline

MCNP6 Development Practices

- Source Code Management and Branching
- Issue Tracking and Merge Requests
- Release Documentation and Reporting

Software Quality Assurance

- General Overview
- MCNP6 Development and NQA-1
- SQA Summary

MCNP6 Resources for Quality Assurance Programs

- User Support
- Testing Support and Tools

Summary and Discussion

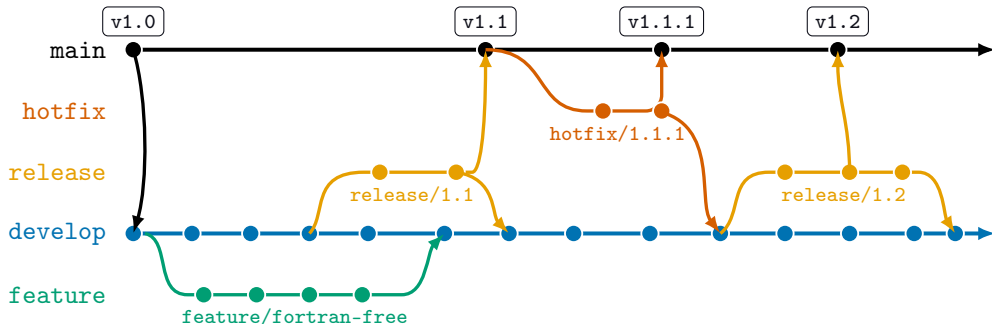
MCNP6 Development Practices

Source Code Management (SCM) with Git

- ▶ All products are maintained under version control using [Git](#) within various repositories. These include:
 - ▶ MCNP6
 - ▶ Regression Tests
 - ▶ Data
 - ▶ Documents
 - ▶ Website & Class Materials
 - ▶ V&V Benchmarks
 - ▶ Python MCNP
 - ▶ Whisper
 - ▶ ISC
 - ▶ MCNPTools
 - ▶ CGM / CGMF
 - ▶ Cinder
 - ▶ Other libraries (turbotally, F5, etc.)
 - ▶ Developer tools and scripts
- ▶ The source code management system primarily used is [GitLab](#).
- ▶ Note that all of these source code repositories the team develops and maintains are hosted on a restricted server allowing export controlled information.

Branching Strategy (Gitflow)

All code changes (merge requests) into most branches (feature branches excluded) requires minimum of one reviewer approval and for code changes to pass continuous integration / continuous deployment (CI/CD) testing.



- **main** Production branch; code should remain deployable.
- **develop** Integration branch for completed features.
- **feature** Created from and merged back into **develop**.
- **release** Created from **develop**; merged into **main** and **develop**.
- **hotfix** Created from **main**; merged into **main** and **develop**.

Issue Tracking in GitLab (1/3)

- ▶ “Work Items” (or Issues) in GitLab contain descriptions of new features, improvements, bugfixes, and maintenance activities.
- ▶ **All** changes to MCNP6 must be associated with a GitLab work item.

Sample
GitLab
Work Item

The screenshot shows a GitLab issue page for 'MCNP6 / mcnp6 / Work items #54340'. The issue is titled 'Runtime cleanup and reorganization' and is in the 'Open' state, created 5 days ago by Michael Evan Rising. The summary states: 'With the new HDF5 runtime, as of MCNP6.3, some more organization and cleanup of this file would be nice to have to allow for improved runtime compatibility and usability support. This issue is meant to: 1. define a path forward toward a more organized runtime 2. iterate on the design with the team 3. provide a parent issue that can be targeted for a specific release, where smaller issues can be created, worked on, and closed to break the larger issue into smaller work chunks.' The issue is categorized under 'MCNP6.3.0 runtime layout' and includes a tree view of the file structure: 'runtime.h5' containing 'config_control', 'fixed', 'header', 'problem_info', 'restart (optional)', 'results (optional)', and 'variable'. The right sidebar shows metadata: Status (To do), Assignees (Michael Evan Rising), Labels (OpenImprovement, Modernization), Parent (None), Weight (None), Milestone (Future), Iteration (None), Dates (Start: None, Due: None), Time tracking (Add an estimate or time spent), and 3 Participants.

MCNP6 / mcnp6 / Work items #54340

Runtime cleanup and reorganization

Open Issue created 5 days ago by Michael Evan Rising

Summary

With the new HDF5 runtime, as of MCNP6.3, some more organization and cleanup of this file would be nice to have to allow for improved runtime compatibility and usability support.

This issue is meant to

1. define a path forward toward a more organized runtime
2. iterate on the design with the team
3. provide a parent issue that can be targeted for a specific release, where smaller issues can be created, worked on, and closed to break the larger issue into smaller work chunks.

MCNP6.3.0 runtime layout

Largely ported from the MCNP6.2 and earlier runtime layout, the layout at the time of the MCNP6.3.0 release looked like:

```
runtime.h5
├── [6] config_control
├── [6] fixed
├── [6] header
├── [6] problem_info
├── [6] restart (optional)
├── [6] results (optional)
├── [6] variable
└── [6] - group
```

The `config_control` group was added specifically to MCNP6.3.0 to handle compatibility and version control of the HDF5 file with versions of MCNP6. This should remain going forward.

The `fixed` group contained fixed data that would remain constant throughout the simulation, written only once.

The `header` group contained specific information about the run.

The `problem_info` group contains a few items related to the problem. This group was also new in MCNP6.3.0, and it seems it was a partial port of some of the `header` information. It seems reasonable to consider moving the `header` items into the `problem_info` group with improved variable names.

The `restart` group contains information needed for a restart/continue run. This was also new in the MCNP6.3.0 release and only includes an `unstructured_mesh` group at the time of that release.

The `results` group contains information organized nicely for quick extraction and access to tally results and other relevant results. This group is new to MCNP6.3.0 and has grown over time to add more helpful results. This is not needed for restarts and should continue to grow with future information.

Read more

Status: To do Edit

Assignees: Michael Evan Rising Edit

Labels: OpenImprovement Modernization Edit

Parent: None Edit

Weight: None Edit

Milestone: Future Edit

Iteration: None Edit

Dates: Start: None Due: None Edit

Time tracking: Add an estimate or time spent. +

3 Participants

Issue Tracking in GitLab (2/3)

- ▶ New features or major modernization efforts can use work items to develop a (re)design of code capabilities.
- ▶ Improvement and/or bugfix work items brought to the team by the community or found by the developers contain,
 - ▶ description of code (mis)behavior,
 - ▶ reproducers of the current and/or intended behavior,
 - ▶ and a plan of action to improve behavior.
- ▶ Maintenance efforts include similar information depending on work scope.
- ▶ Commits contain work item description and implementation information.

Sample Git Commit Message

```
commit ad61005bb8b51682834e97f088819c8e7f8dc8c
Author: Michael Aaron Lively <livelym@lanl.gov>
Date: Tue Jan 6 15:46:14 2026 -0700

    Fix energy deposition for Compton-induced relaxation

    Closes #53960

    This fix yields (close to) correct energy deposition and yields correct peaks in pulse-height tally spectra when using Compton-induced fluorescence with EPRDATA libraries.

    * Subtract the sampled Compton-induced relaxation subshell binding energy from the incident photon energy.
    * Apply this correction to Doppler broadening.
    * Fix energy deposition for Compton-induced relaxation.
    * Fix test results to reflect code changes.
    * Add regression tests for these code changes.
    * Update release notes.
    * Change single-event electrons PTRAC test to reduce file size.

    Notes:
    * The fix in this MR essentially subtracts the sampled subshell binding energy for the Compton electron from the incident photon energy before computing the incoherent scattering kinematics. This is probably not physically correct, but (a) it is *more* physically correct than the prior implementation, and (b) we lack the tabulated data needed to do anything more accurate.
    * A minor side effect from this change is a *small* reduction in the total generation of secondary particles, since some energy is now (correctly) lost to overcome the electron binding energy. Since fewer particles are transported (subject to randomness, of course), this means that there is a very minor reduction in runtime due to these changes.
```

Issue Tracking in GitLab (3/3)

- ▶ The manual, release notes, V&V report, and build guide documents under version control as well.
 - ▶ All exist within the same MCNP6 repository.
 - ▶ Allows for code and documentation to remain consistent at all times.
- ▶ Resolved and known issues are described in the appendices of the release notes document.

Sample MCNP6.3.1 Issues in Release Notes Appendix

A MCNP6.3.1 Specific Activities

A.1 New Features

MCNP-53464	A new dentype keyword is added to the EMBED card and is required to embed LNK3DNT files [§2.1].
MCNP-53671	A new random number generator is added [§2.3].

A.2 Code Enhancements

A.2.1 Data and Physics

MCNP-53601	Update the xsdtr format and add named identifier support to the code [§2.2].
MCNP-53626	The tabular data expung , F90 routine had a limit of reading 3000 reactions. This limitation is removed, allowing for arbitrary quantities of reactions.
MCNP-53627	The MARS physics routines had a hardcoded limit of allowed (Z, A) pairs in a simulation of 500. This limitation is removed.
MCNP-53844	The dbrc_make_lib utility, which generates data needed for the DBRC card, is modified to add in the necessary information to generate the DBRC-endf81.txt file.

A.2.2 Unstructured Mesh

MCNP-59954, MCNP-59569

Merge Requests in GitLab (1/2)

- ▶ “Merge Requests” in GitLab allow the team to evaluate proposed code changes that resolve a given work item.
- ▶ The merge request author works with the reviewer(s) to ensure code changes
 - ▶ appropriately address the work item,
 - ▶ do not introduce unintended behavior,
 - ▶ and conform to the coding standards the team uses.

Sample GitLab Merge Request

The screenshot shows a GitLab Merge Request (MR) titled "Add spherical FMESH tracking support". The MR is created by Colin Andrew Weaver and is targeting the "dev" branch. It has 108 changes, 39 commits, and 15 pipelines. The MR is currently closed (#54271). The description of the MR includes a summary of the changes and links to the relevant code. The MR is assigned to Colin Andrew Weaver and has two reviewers: Jeffrey S. Bull and Colin James Josey. The MR is labeled "Type Improvement" and has a milestone of 6.4. The MR is also linked to a work item titled "Add spherical FMESH tracking support". The MR is created 1 month ago by Colin Andrew Weaver.

Add spherical FMESH tracking support

Colin Andrew Weaver requested to merge [cawea/spherical-fmesh](#) into [dev](#) 4 months ago

Overview 107 Commits 39 Pipelines 15 Changes 108

#54271 (closed)

Adds FMESH tracking support for particle tallying in a spherical structured mesh.

Tracking

The spherical tracking algorithm was copied from `variance_reduction/weight_along_surweg.F90` into the `mesh_score_sph` subroutine. `mesh_score_sph` is located in `fmesh/transport_ssd.F90` and follows the same flow as `mesh_score_cyl`.

Plotting

Spherical FMESH plotting uses the coordinates of the pixel center to find the mesh bin and set pixel colors accordingly. The code for the spherical FMESH wire frame was copied from `plot/plotmesh.F90`.

Testing

Adds 13 regression problems (78 tests) in `Testing/features/mesh_tally/spherical/`:

- Vacuum test verifying geometric attenuation - [inp61](#)
- Origin-crossing tests (axial and oblique paths) - [inp62](#) through [inp68](#)
- Homogeneous sphere test comparing spherical FMESH against F2 / F4 tallies - [inp69](#)
- Surface current tally - [inp70](#)
- Spherical FMESH to F4 comparison - [inp11](#)
- Spherical FMESH with `type = scalar` for a uniform source in a sphere - [inp12](#)
- Spherical FMESH to F4 comparison - [inp13](#)

Edited 1 month ago by Colin Andrew Weaver

Assignee Edit
Colin Andrew Weaver

2 Reviewers Edit
All required approvals given Assign
Jeffrey S. Bull
Colin James Josey

Labels Edit
[Type Improvement](#)

Milestone Edit
6.4

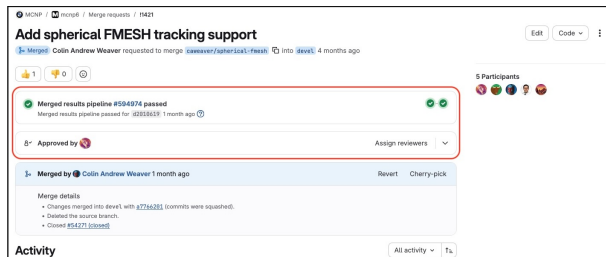
Work Items
Closing
Add spherical FMESH tracking support

Time tracking +
No estimate or time spent

Merge Requests in GitLab (2/2)

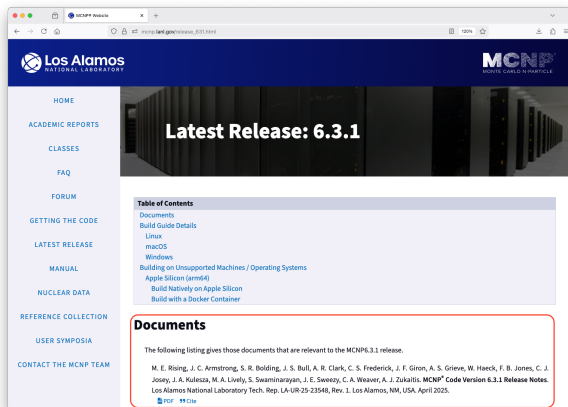
- ▶ Continuous integration and continuous deployment (CI/CD) is used to automatically perform
 - ▶ checks that work item is documented in commit message,
 - ▶ static analysis of code formatting consistency,
 - ▶ and regression testing of code changes with various configurations to ensure new behavior is verified.
- ▶ GitLab pipelines used to perform the automatic checks and tests.
- ▶ Minimum of a single reviewer must approve before merge.

Sample
Pipeline and
Approval Status



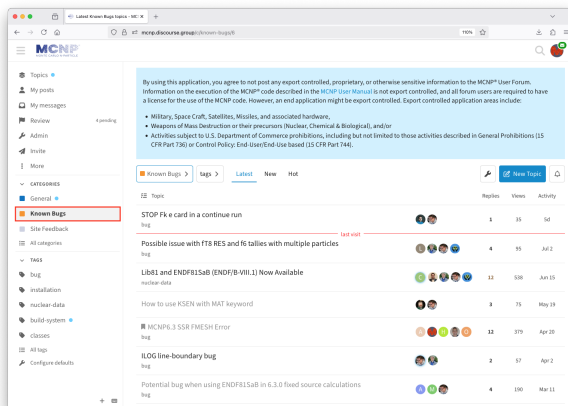
Release Documentation

- ▶ Release page updates ([MCNP6.3.1 Release Webpage](#))
 - ▶ Release notes ([LA-UR-25-23548, Rev.1](#))
 - ▶ User and theory manual ([LA-UR-24-24602, Rev.1](#))
 - ▶ Build guide ([LA-UR-25-22869, Rev.1](#))
 - ▶ V&V report ([LA-UR-25-22398, Rev.1](#))
- ▶ Reminder: the release notes documents include both resolved and known issues at the time of that release.



Website Updates and Reporting

- ▶ Website includes additional information, such as
 - ▶ FAQs and an extensive reference collection.
 - ▶ Additional release-specific build information for third-party dependencies and tips for unsupported build environments.
- ▶ The **new MCNP forum** includes up-to-date information with a specific category of *known bugs* with community and developer discussion.



Software Quality Assurance

General Overview (1/3)

- ▶ For more information on the Software Quality Assurance (SQA) of the MCNP code, see the [MCNP reference collection SQA](#) section on the website.
- ▶ Specifically, this presentation largely covers the information provided in the [most recent SQA document](#) available.
- ▶ The current hierarchy of SQA authority at LANL, as it pertains to MCNP6, is as follows:
 - ▶ The MCNP6 code is developed in the XCP-3 Monte Carlo Codes group at LANL within the X Computational Physics Division (XCP).
 - ▶ Within the Associate Laboratory Directorate for Weapons Physics (ALDX), the Weapons Research Services Secure Networks and Assurance Group (WRS-SNA) manages the SQA plan for the XCP division to meet the broader LANL SQA requirements.

General Overview (2/3)

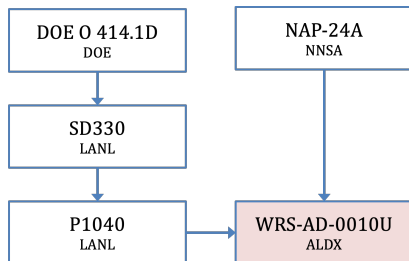
- ▶ Each XCP software product is categorized into one of three software types:
 - ▶ Safety Software
 - ▶ Non-Safety Risk Significant Software
 - ▶ Non-Safety Commercially Controlled Software
- ▶ Within ALDX, all software that is:
 - ▶ Categorized as Safety Software and Non-Safety Risk Significant Software: the *LANL P1040 Software Quality Management* plan is followed
 - ▶ Categorized as Non-Safety Commercially Controlled Software: the *WRS-SNA WRS-AD-0010U Software Quality Assurance Program Plan* is followed.

Note: The NNSA NAP-24A Weapon Quality Policy requires that a standard be selected. WRS-AD-0010U identifies Institute of Electrical and Electronics Engineers (IEEE) as its standard.

General Overview (3/3)

- ▶ In 2018, the MCNP6 code at LANL was categorized by the XCP division as **Non-Safety Commercially Controlled Software**. This categorization ultimately determines which SQA plan to follow.
- ▶ Additionally, the MCNP6 code was graded as **Medium Impact** software product. The grade ultimately determines which components of the SQA plan are required to be followed.

SQA Document Hierarchy



SQA Requirements Determination

Procedural Scope by Software Type				
Safety Software	Non-Safety <i>Risk Significant Software</i>	Non-Safety <i>Commercially Controlled Software</i> (includes all ALDX in-scope software)		
Follow P1040		High Impact	Medium Impact	Low Impact
Follow WRS-AD-0010U				

MCNP6 Development and NQA-1 (1/5)

- ▶ NQA-1: “Quality Assurance Requirements for Nuclear Facility Applications”
 - ▶ Provides a comprehensive framework for quality in nuclear-related activities, including software development.
- ▶ For MCNP development,
 - ▶ The code can be used for many applications ranging from radiation protection and shielding to medical physics to fission and fusion reactor design to nuclear criticality safety.
 - ▶ Unfortunately, given the development history and age of the code, not all capabilities were historically developed under rigorous SQA standards, including NQA-1.
 - ▶ Currently, the MCNP code development team adheres to the majority of the software engineering practices aligned with NQA-1.
 - ▶ Often, based on safety significance and complexity of software components, a graded approach may be applied.

MCNP6 Development and NQA-1 (2/5)

Summary of NQA-1 Requirements and Relevant MCNP Development Practices

NQA-1 Requirement	Applicability	MCNP Development Practice
Requirement 1 - Organization	Full	The MCNP development team has a defined organizational structure with clear roles and responsibilities for SQA, configuration management, technical team leadership, and development leads.
Requirement 2 - Quality Assurance Program	Full	A documented software quality assurance plan, based on the prescribed risk categorization and impact rating, exists for MCNP development.
Requirement 3 - Design Control	Graded	For high-impact components (e.g., capabilities associated with criticality safety, large-scale refactor of a core capability), the design and plans are documented. Lower-impact components follow less rigorous design practices.
Requirement 4 - Procurement Document Control	Graded	Procurement of external software tools or services is controlled through approval processes. For open-source or commercial components used within the code, the license implications, risk, and long-term stability are assessed prior to procurement and adoption.
Requirement 5 - Instructions, Procedures, and Drawings	Graded	Development procedures are documented and controlled. Procedures include coding standards and best practices, code review guidelines, code testing procedures, and build instructions.

MCNP6 Development and NQA-1 (3/5)

Summary of NQA-1 Requirements and Relevant MCNP Development Practices

NQA-1 Requirement	Applicability	MCNP Development Practice
Requirement 6 - Document Control	Graded	All project documents are stored in a version-controlled repository (e.g., GitLab) with access controls, review workflows, and release tagging.
Requirement 7 - Control of Purchased Items and Services	Graded	Similar to Requirement 4, open-source or commercial components (e.g., compilers, third-party libraries) are evaluated for suitability.
Requirement 8 - Identification and Control of Items	N/A	Physical items requiring identification and control are not typically needed for software development.
Requirement 9 - Control of Special Processes	N/A	Special processes are not typically involved in software development.
Requirement 10 - Inspection	N/A	Software inspection/verification is performed through reviews, testing, and analysis, covered under other requirements. For example, see Requirements 3 and 11.

MCNP6 Development and NQA-1 (4/5)

Summary of NQA-1 Requirements and Relevant MCNP Development Practices

NQA-1 Requirement	Applicability	MCNP Development Practice
Requirement 11 - Test Control	Graded	Varied testing approach: unit, integration, system, and regression testing for entire code with automated testing to ensure behavior. Verification and validation testing based on safety significance, and historical use and robustness of capability (e.g., criticality safety applications heavily verified and validated). Verification and validation results documented for specific applications.
Requirement 12 - Control of Measuring and Test Equipment	N/A	See Requirement 4.
Requirement 13 - Handling, Storage, and Shipping	N/A	No physical handling, storage, or shipping of items is required for software development.
Requirement 14 - Inspection, Test, and Operating Status	N/A	No monitoring of the status of physical items involved in software development.
Requirement 15 - Control of Nonconforming Items	N/A	Issue tracking and resolution for software development applied in Requirement 16.

MCNP6 Development and NQA-1 (5/5)

Summary of NQA-1 Requirements and Relevant MCNP Development Practices

NQA-1 Requirement	Applicability	MCNP Development Practice
Requirement 16 - Corrective Action	Full	Corrective action process for bugs, defects, and misbehaviors. Work items or issues created in tracker (i.e., GitLab work items), prioritized, investigated, and resolved. New or updated tests exhibit the improved behavior.
Requirement 17 - Quality Assurance Records	Full	QA records and documents maintained in version-controlled repository.
Requirement 18 - Audits	Graded	Internal audits and assessments of software development process conducted periodically.
Subpart 2.7 - Software Quality Assurance	Graded	SQA activities throughout life cycle (design, implementation, review, testing); SQA graded per application area.
Subpart 2.14 - Commercial Grade Dedication	Graded	Rarely, if ever, has commercial-grade dedication been needed for MCNP development.

For all intents and purposes, the current MCNP6 development practices are aligned with the intent of the NQA-1 standards.

SQA Summary for Nuclear Criticality Safety Users

The primary codes developed with support from the DOE Nuclear Criticality Safety Program (NCSP) can be used in NCS applications given they follow their *site-specific SQA program for classifying and controlling software*.

From DOE-STD-3007-2017 on Preparing Criticality Safety Evaluations at Department of Energy Nonreactor Nuclear Facilities:

“The neutron transport computer code systems listed below are developed and maintained through rigorous expert review of neutron transport theory, cross section data, and Monte Carlo methods in accordance with DOE software quality assurance requirements. These code systems are distributed by the Radiation Safety Information Computational Center at Oak Ridge National Laboratory. The following code systems are accepted programs for use in NCS applications when used in accordance with a site-specific software quality assurance program for classifying and controlling software:

- ▶ *SCALE: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design*
- ▶ *MCNP: Monte Carlo N-Particle Transport Code System*
- ▶ *COG: Multiparticle Monte Carlo Code System for Shielding and Criticality Use*

All pertinent calculational results shall be reported...”

SQA Summary for General Users

- ▶ For the applications and uses of versions of the MCNP6 code that are developed and released by the Monte Carlo Codes group (XCP-3), the MCNP6 code is not controlled for and should not be used for safety significant applications unless qualified to do so by individual users of the code for their specific areas of application.
 - ▶ Note this also includes LANL. If MCNP6 is deemed to be needed for safety and/or risk significant applications, its use must follow the P1040 Software Quality Management process.
- ▶ From the development team perspective, we try to adhere to best SQA practices throughout the development process, and then provide robust software, documentation, tools, and support to the community of users.

MCNP6 Resources for Quality Assurance Programs

Installation Support

- ▶ Pre-compiled binaries (see release notes document) provided, including:
 - ▶ Standard builds for Windows, Linux, and macOS.
 - ▶ MPI-compatible builds for Windows, Linux (OpenMPI and MPICH), and macOS.
- ▶ For those with source code,
 - ▶ Build guidance provided in release documentation (see build guide) and on release page of website.
 - ▶ Regression testing provided if needing to verify behavior of modified code version or non-standard build environment.
- ▶ Simple installation testing script to check environment, data, and basic code execution.
- ▶ Installation videos publicly available to help guide through the installation process.

User Support and Feedback Loop

- ▶ Users can ask questions and report issues to the [MCNP team help system](#).
 - ▶ Issues saved in repository via GitLab issue tracker.
 - ▶ Entire team of experts can help resolve issues.
 - ▶ Bugs submitted are migrated and prioritized in the general MCNP6 issue tracker.
- ▶ Community discussions and contributions on the new MCNP forum.
 - ▶ Active community of new and experienced MCNP users.
 - ▶ The development team participates in discussion and issue resolution.
- ▶ Regular user workshops and training/class sessions available throughout the year.

Testing Support and Tools (for NCS) (1/2)

- ▶ For NCS use, verification and validation of the computer code system is required.
 - ▶ This is typically done upon installation and then periodic testing is performed to ensure computer code system remains reliable.
- ▶ Verification and validation suites are provided with MCNP6, which may be used as is or to augment NCS verification and validation needs.

Verification

Suite	Contents	Use Case(s)
regression	1000's of behavioral tests confirming the code functionality remains consistent from version to version	developer use on a daily basis; only needed by user if code is modified
keff	75 analytical k_{eff} benchmarks in both multigroup and continuous-energy form; testing Monte Carlo methods work as expected	for production code testing by developers or installation/periodic NCS testing

Testing Support and Tools (for NCS) (2/2)

Validation		
Suite	Contents	Use Case(s)
criticality	31 International Criticality Safety Benchmark Evaluation Project (ICSBEP) handbook models	for production code testing by developers or installation/periodic NCS testing
crit_expanded	119 ICSBEP handbook models covering a variety of configurations	
Whisper / Los Alamos Benchmark Suite (LABS)	1101 ICSBEP handbook models undergoing quality control review and revision procedures	used by the Whisper code for NCS validation

These test suites, especially the `keff` verification and the criticality validation suites, make use of a standalone testing framework that can be used to regularly exercise these tests for NCS computer code system monitoring, qualification, and validation.

Summary and Discussion

Summary and Discussion

- ▶ The current MCNP6 code development process uses modern tools, and follows standard software engineering and development practices.
- ▶ We aim to follow best SQA practices throughout the development process so that we are able to provide robust software, clear documentation, and user-friendly tools to the broader community of users.
- ▶ Of course, we want to hear if there are things (within our control) that can be done to make qualification and validation of the code easier for the user community.

Acknowledgements

This work is supported by the Department of Energy (DOE) Nuclear Criticality Safety Program.

This work is supported by the Department of Energy through Los Alamos National Laboratory (LANL) operated by Triad National Security, LLC, for the National Nuclear Security Administration (NNSA) under Contract No. 89233218CNA000001.

Questions?

mrising@lanl.gov