|  |
| --- |
| Document ID: PLN-ID\_5553  Revision ID: 0 REV  Effective Date: TBD |
| Configuration Management Plan |
|  |
|  |
|  |
|  |
| RAVEN and RAVEN Plug-ins Software Configuration Management Plan |

|  |  |  |
| --- | --- | --- |
| RAVEN and RAVEN Plug-ins Software Configuration Management Plan | | |
| PLN-5553 | | |
|  | | |
| **Prepared by:** |  | |
| [Required. For EDMS controlled documents, reference the eCR.] |  |  |
| Technical Lead |  | Date |
| **Reviewed by:** |  | |
| [Required for QL-1 and QL-2. For EDMS controlled documents, reference the eCR.] |  |  |
| Independent Reviewer/Review Team |  | Date |
| **Approved by:** |  | |
| [Required. For EDMS controlled documents, reference the eCR.] |  |  |
| IT Asset Owner |  | Date |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Applicability: | Plan |  | eCR Number: |  |
| Manual: | | | | |

CONTENTS

[1. INTRODUCTION 4](#_Toc526071943)

[1.1 Purpose 4](#_Toc526071944)

[1.2 Scope 4](#_Toc526071945)

[1.3 Limitations and Assumptions 4](#_Toc526071946)

[2. REFERENCES 5](#_Toc526071947)

[3. DEFINITIONS AND ACRONYMS 6](#_Toc526071948)

[3.1 Definitions 6](#_Toc526071949)

[3.2 Acronyms 11](#_Toc526071950)

[4. MANAGEMENT 12](#_Toc526071951)

[4.1 Roles and Responsibilities 12](#_Toc526071952)

[4.2 Applicable Policies, Directives, and Procedures 12](#_Toc526071953)

[5. CM ACTIVITIES 12](#_Toc526071954)

[5.1 Configuration Identification 13](#_Toc526071955)

[5.1.1 Identifying Configuration Items 13](#_Toc526071956)

[5.1.2 Naming Configuration Items 13](#_Toc526071957)

[5.1.3 Acquiring Configuration Items 13](#_Toc526071958)

[5.2 Configuration Control 14](#_Toc526071959)

[5.2.1 Requesting Changes 15](#_Toc526071960)

[5.2.2 Evaluating Changes 16](#_Toc526071961)

[5.2.3 Approving or Disapproving Changes 18](#_Toc526071962)

[5.2.4 Implementing Changes 19](#_Toc526071963)

[5.3 Configuration Status Accounting 21](#_Toc526071964)

[5.4 Configuration Evaluation and Reviews 22](#_Toc526071965)

[5.5 Subcontractor/Vendor Control 22](#_Toc526071966)

[6. CM Schedules 23](#_Toc526071967)

[7. CM Resources 24](#_Toc526071968)

[8. Plan Maintenance 25](#_Toc526071969)

# INTRODUCTION

## Purpose

The purpose of this configuration management plan (CMP) is to document the configuration management (CM) activities, plan management, and maintenance needed to assure proper configuration of the Risk Analysis and Virtual ENvironment (RAVEN) software and its supported Plug-ins.

*Configuration Management* (see def.) consists of activities to control and manage changes to items that have a *baseline* (see def.). It includes the process of identifying the *configuration items* (CIs) (see def.) in a system, controlling the release and change of these items, and recording and reporting the status of the CIs and their associated change requests.

## Scope

This CMP includes the organization, roles, responsibilities, policies, directives, and procedures needed to manage the RAVEN application throughout its lifetime. Specifically, it outlines the configuration identification, controls, status accounting, evaluation, and reviews. It also plans interface and how the application will be validated, released, and delivered.

## Limitations and Assumptions

* All software management practices must be in compliance with LWP-13620, “Managing Information Technology Assets”, including all software management activities performed for Quality Level (QL)-1, QL-2, and QL-3 application software and associated support software (see def.). This plan applies directly to custom-developed software (see def.) and acquired applications and support software managed by the modeling and simulation team.
* The modeling and simulation team will adhere to LWP-1303, “Management of Unclassified Cyber Security Information Systems” and LWP-1401, “Preparing and Releasing Scientific and Technical Information Products,” where applicable.
* 29 USC 794d, Section 508 of the Workforce Investment Act of 1998 considerations will be made for the ability of disabled individuals to access the information or service provided by the software.
* Adequate funding, required hardware, and system software is available to complete planned RAVEN application activities.
* Roles and responsibilities cited in this plan can be reassigned as needed by the project manager or personnel designated by the information technology (IT) asset owner.
* INL will manage the software with support from vendors (for *acquired software* [see def.]) until the software is retired.
* Software vendor support agreements are maintained.
* For firmware, changes to acquired software including software updates and security patches will be implemented by the product vendor.
* All changes to safety software (see def.) including security patches will be controlled through the change control board (see def., CCB). For external release, the project or maintenance and operations (M&O) team will adhere to LWP-1401, “Preparing and Releasing Scientific and Technical Information Products.”
* The software will be configured for multiple users via the INL network as allowed by the software.
* RAVEN supports retention of quality records in compliance with LWP-1202, “Records Management.”

# REFERENCES

The following are references for this CMP.

ASME NQA‑1‑2008 with the NQA-1a-2009 addenda, “Quality Assurance Requirements for Nuclear Facility Applications”

DOE O 414.1D, “Quality Assurance”

ISO/IEC/IEEE 24765:2010(E), “Systems and software engineering — Vocabulary”

LWP‑1201, “Document Management”

LWP-1202, “Records Management Plan”

LWP-1306, “Management of IT Asset Minimum Security Configurations,” Rev. 1, December 23, 2013.

LWP‑13620, “Managing Information Technology Assets”

PLN-5552, “RAVEN Software and RAVEN Plug-ins Quality Assurance Plan (SQAP)

# DEFINITIONS AND ACRONYMS

This section defines, or provides the definition of, all terms and acronyms required to properly understand this plan.

## Definitions

*Acquired software.* Software generally supplied through basic procurements, two-party agreements, or other contractual arrangements. Acquired software includes commercial off-the-shelf software, support software such as operating systems, database management systems, compilers, software development tools, and commercial calculational software and spreadsheet tools (e.g., Mathsoft’s MathCad and Microsoft’s Excel). Downloadable software that is available at no cost to the user (referred to as freeware) is also considered acquired software. Firmware is acquired software. Firmware is usually provided by a hardware supplier through the procurement process and cannot be modified after receipt.

*Anomaly.* Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents.

*Baseline*. A specification or product (e.g., project plan, maintenance and operations [M&O] plan, requirements, or design) that has been formally reviewed and agreed upon, that thereafter serves as the basis for use and further development, and that can be changed only by using an approved change control process. [ASME NQA-1-2008 with the NQA-1a-2009 addenda edited]

*Change control.* An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items (CIs see def.) after formal establishment of their configuration identification. [ISO/IEC/IEEE 24765:2010(E)]

*Change control board (CCB).* The group by which a change is proposed, evaluated, approved or rejected, scheduled, and tracked. This board is also responsible for evaluating and approving or disapproving proposed changes to configuration items (CIs) and implementation of approved changes when required.

*Change requests (CRs).* CRs can be initiated by anyone, including off site users, and can be used for maintenance (fine-tuning and problem resolving), new development, and enhancements, or can be used to report program errors and problems.

*Change request log.* A log that provides a listing of all the change requests and the change request status used for application software, system software, and hardware configuration control.

*Configuration Control*. An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. [ISO/IEC/IEEE 24765:2010(E)]

*Configuration identification.* An element of configuration management, consisting of selecting the configuration items (see def.) for a system and recording their functional and physical characteristics in technical documentation.

*Configuration Item*. An item or aggregation of hardware or software (including documentation) or both that is designed to be managed as a single entity. [ISO/IEC/IEEE 24765:2010(E) edited]

*Configuration Management*. A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a *configuration item* (see def.), control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. [ISO/IEC/IEEE 24765:2010(E)]

*Custom-built IT assets.* Information technology (IT) assets designed, developed, or modified internally or by a qualified subcontractor through the procurement process. Examples include custom-developed (see def.) or customized software, spreadsheet, and calculation and analysis applications (e.g., computer models), the implementation of a new network infrastructure or IT technology (e.g., Gmail, Internet Protocol Version 6, Internet Explorer 9). [Developed for internal laboratory use]

*Custom-developed software.* Software built specifically for a DOE application or to support the same function for a related government organization. It may be developed by DOE or one of its M&O contractors or contracted with a qualified software company through the procurement process. Examples of custom-developed software include material inventory and tracking database applications, accident consequence applications, control system applications, and embedded custom-developed software that controls a hardware device.

*Defect.* An error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

*Doxygen*. Standard tool for generating documentation from annotated C++ sources.

*Electronic Document Management System (EDMS).* System approved for long-term storage, management, and maintenance of electronic and hardcopy records.

*Enterprise Architecture (EA) Repository.* An Oracle database that houses information about software applications and servers, and is the source for the INL data dictionary. The applications are related to the management system business functions it supports or implements. EA is the repository for the technology (e.g., software/hardware) used to construct and implement software applications. EA contains links to the software documentation stored in EDMS (see def.) and includes a list of software owners.

*GitHub.* A web-based revision control hosting service for software development and code sharing. GitHub provides additional tools such as documentation generation, issue tracking, Wikis, nested task-lists within files, etc.

*GitLab.* A web-based revision control hosting service for software development and code sharing similar to GitHub. GitLab is used for the applications built on the MOOSE framework. MooseBuild connects to both the external and internal GitHub/GitLab to perform software builds.

*Issue.* Issues can be initiated by anyone, including off site users, and are used for maintenance (fine-tuning and problem resolving), new development, enhancements, or can be used to report program errors and problems.

*Issue (GitHub).* As defined for the GitHub environment, issues are suggested improvements, tasks, or questions related to the repository. Issues can be created by anyone (for public repositories) and are moderated by repository collaborators. Each issue contains its own discussion forum and can be labeled and assigned to a user.

*Major Change*. A revision to software that, in the best judgment of authorizing personn*el*, has the potential to compromise the accuracy/validity of the output data, and as a result, could diminish the margin of safety to the public, worker, or environment.

*Method.* A reasonably complete set of rules and criteria that establish a precise and repeatable way of performing a task and arriving at a desired result. [The Configuration Management Manual Guideline for Improving the Software Process, Carnegie Mellon University Software Engineering Institute, 1995]

*Minor Change.* A revision to software that, in the best judgment of authorizing personnel, will not compromise the accuracy/validity of the output data and will not diminish the margin of safety to the public, worker, or environment.

*Open source.* Denoting software for which the original source code is made freely available and may be redistributed and modified.

*Pull requests.* Pull requests can be initiated by anyone, including off-site users, and are used for maintenance (fine-tuning and problem resolving), new development, enhancements, or can be used to report program errors and problems. Pull requests let you tell others about changes you have pushed to a repository on GitHub. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.

*Regression testing.* Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

*Safety function.* The performance of an item or service necessary to achieve safe, reliable, and effective utilization of nuclear energy and nuclear material processing. For INL, safety functions are identified and defined in a formal safety basis or commitment document as credited for achieving nuclear safety (e.g., safety structures, systems, and components; safety significant; safety class; safety related; or important to safety) (ASME NQA-1-2008 with the NQA-1a-2009 addenda edited).

*Safety software.* Software, including the following:

* **Safety system software.** Software for a nuclear facility that performs a safety function as part of a structure, system, or component and is cited in either (a) a DOE-approved documented safety analysis, or (b) an approved hazard analysis per DOE P 450.4, “Safety Management System Policy,” dated October 15, 1996 (or latest version) and 48 CFR 970-5223.1, “Integration of Environment, Safety, and Health into Work Planning and Execution.”
* **Safety analysis and design software.** Software that is used to classify, design, or analyze nuclear facilities. This software is not part of a structure, system, or component, but helps to ensure that proper accident or hazards analysis of nuclear facilities or a structure, system, or component that performs a safety function.
* **Safety management and administrative controls software.** Software that performs a hazard control function in support of nuclear facility or radiological safety management programs or technical safety requirements or other software that performs a control function necessary to provide adequate protection from nuclear facility or radiological hazards. This software supports eliminating, limiting, or mitigating nuclear hazards to worker, the public, or the environment as addressed in 10 CFR Parts 830, 835, the DEAR ISMS (Integrated Safety Management System) clause, and 48 CFR 970‑5223.1.

*Software.* Computer programs and associated documentation and data pertaining to the operation of a computer system and includes application software and support software.

*Software life cycle.* The activities that comprise evolution of software from conception to retirement. The software life cycle typically includes the activities associated with requirements, design, implementation, test, installation, operation, maintenance, and retirement.

*Software quality assurance.* All actions that provide adequate confidence that software quality is achieved.

*Software tool*. A computer program used in development, testing, analysis, or maintenance of a program or its documentation. Examples include comparators, cross-reference generators, compilers, computer-aided software-engineering tools, configuration and code management software, decompilers, disassemblers, editors, flowcharters, monitor test case generators, and timing analyzers.

*Support software.* Software tools (see def.) and system software (see def.).

*System software.* Software designed to facilitate operation and maintenance of a computer system and its associated programs (e.g., operating systems, assemblers, and utilities).

*System testing.* Testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements.

*Task (GitHub).* A suggested improvement or feature enhancement.

*Test case. (1)* A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.

*Test driven development.* A method of software development in which unit testing is repeatedly conducted on source code. After each test, refactoring is done and the same or a similar test is performed again. The process is iterated until the unit functions in accordance with the specifications.

*User documentation.* Instructions for use describing the capabilities and intended use of the software within specified limits. May also include a theory manual, when relevant.

*Validation.* Confirmation, through the provision of objective evidence (e.g., acceptance test), that the requirements for a specific intended use or application have been fulfilled. [ISO/IEC/IEEE 24765:2010(E) edited]

*Verification.* (1) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (2) Formal proof of program correctness (e.g., requirements, design, implementation reviews, system tests). [ISO/IEC/IEEE 24765:2010(E) edited]

## Acronyms

ASME American Society of Mechanical Engineers

CCB change control board

CI Configuration Item

CM Configuration Management

CMP Configuration Management Plan

CR Change request

DOE Department of Energy

EA Enterprise Architecture

EDMS Electronic Document Management System

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronics Engineers

INL Idaho National Laboratory

ISO International Organization for Standardization

IT Information Technology

M&O Maintenance and Operations

NQA Nuclear Quality Assurance

QA Quality Assurance

QL Quality level

QLD Quality Level Determination

RAVEN Risk Analysis and Virtual Environment

SRS software requirements specification

SSD safety software determination

SQA software quality assurance

SQAP software quality assurance plan

V&V verification and validation

# MANAGEMENT

The project manager or M&O manager is responsible for adhering to management and operations requirements per LWP-13620.

## Roles and Responsibilities

RAVEN core team members are solely responsible for conducting configuration management activities. The same person may hold multiple roles, and responsibilities may be delegated at any point during the life cycle. All roles are assigned and documented in the Asset Portfolio.

The detailed roles and responsibilities are reported in Section 4.3 of PLN-5552, “RAVEN Software and RAVEN Plug-ins Quality Assurance Plan”.

## Applicable Policies, Directives, and Procedures

The INL Software Quality Assurance (SQA) Program is implemented in compliance with DOE O 414.1D, NQA‑1‑2008 with the 2009 addenda, and meets format requirements for IEEE Stds 828‑2012, IEEE Standard for CM in Systems and Software Engineering. Procedures implementing these requirements include LWP‑13620, “Managing Information Technology Assets” and LWP‑1303, “Management of Unclassified Cyber Security Information Systems”. These procedures will be implemented for RAVEN Assets CM.

The set of test cases/procedures are documented and controlled as per LWP-1201, “Document Management,” and LWP-1202, “Records Management.” The test cases/procedures shall be approved prior to the system being approved for use.

# CM ACTIVITIES

Software configuration management activities, including *configuration* *identification* (see def.), *change control* (see def.), status accounting, and software configuration audits, are established during the planning phase of the software life cycle and implemented through operations and maintenance until the product is retired. Configuration items shall be identified by the technical lead in conjunction with the Asset owner and maintained under configuration management until the software is retired.

## Configuration Identification

### Identifying Configuration Items

RAVEN consists of five major components: application software, system software, support software, hardware, and documentation. Each of these items is described in LST-1136. The technical lead identifies these components that are controlled as CIs and will be identified in *the Enterprise Architecture (EA) Repository* (see def.). Individual CIs will be controlled in the separate repositories (GitHub, GitLab) managed by the applicable software team. Controlled CIs shall include documentation (e.g., requirements, design, instructions for use, test plans and test reports), computer programs (source, object, backup files), *support software* (see def.), and hardware. Per LWP-13620, the RAVEN core team is responsible for the ongoing maintenance of CIs associated with their software inventoried in the EA repository. A software baseline that includes the CIs will be established and maintained throughout the software life cycle.

For configuration items, it is not necessary to identify each distinct software file that is to be modified. Instead, an application or module-level CI designation can be supplied to designate the software portion that is being modified. Based on risk, it is at the discretion of RAVEN (and RAVEN Plug-ins) core team technical leads to determine the level of detail for the CI list.

### Naming Configuration Items

All *commercial off-the-shelf (COTS)* (see def.) software items will retain the name given by the vendor. System document deliverables shall be assigned a unique identifier in accordance with LWP-1201, “Document Management.”

All hardware will be controlled according to LWP-2001, “Control of INL Government Property.”

### Acquiring Configuration Items

The Asset owner, with support from the INL Procurement organization, will acquire materials and services that are necessary to support RAVEN software and its supported Plug-ins. These acquisitions include otherwise acquired software (i.e., software that has not been previously approved under a program consistent with the INL Quality Assurance program including freeware, shareware, and firmware).

When new *configurable items (CIs)* (see def.) are acquired, they shall be logged in LST-1136 and the *enterprise architecture (EA)* (see def.) repository will be updated, as needed, to reflect any changes.

## Configuration Control

Changes can be initiated by:

* External or regulatory changes that result in new software requirements
* Internal changes that result in new software requirements or design
* Upgrades for performance, adaptability, etc.
* New technologies that need to be incorporated
* Software refactoring
* Changes in the operating environment
* Reported software problems that must be corrected

The activities outlined in this section shall be followed when any changes are made to RAVEN software and its supported Plug-ins (covered by scope of this plan).

As outlined in PLN-5552, The RAVEN software and its supported Plug-ins operate under a rapid development environment where modifications to the software applications are tracked under a tracking issue.

The change control activities for CI baselines (see def.) consist of requesting a change, an evaluation, an approval or disapproval, notification to requester, design, implementation, acceptance testing, and closure of changes. Changes encompass both error correction and enhancement. The degree of formality necessary for the change process depends on the project baseline affected and on the impact of the change within the configuration structure. Configuration control activities apply to all CIs including documentation, hardware, support software, application software, and the processing of requests for deviations and waivers from the provisions of specifications or acquirer-supplier contracts. For operating system or application software deviations from INL’s [minimum security configuration](https://mypc.inl.gov/) (USGCB MSC), follow the [Computer Security and Operational Variance](https://opscenter.inl.gov/cyber/varinst.pdf) process to identify and obtain approvals for business-necessary variances in accordance with LWP-1306, “Management of IT Asset Minimum Security Configurations.”

The organization’s process for tracking CRs is used for logging activities throughout the change control process. See Figure Figure 1 and Figure 2 for a depiction of the RAVEN core team’s configuration control process.

Configuration baselines are established for each revision.

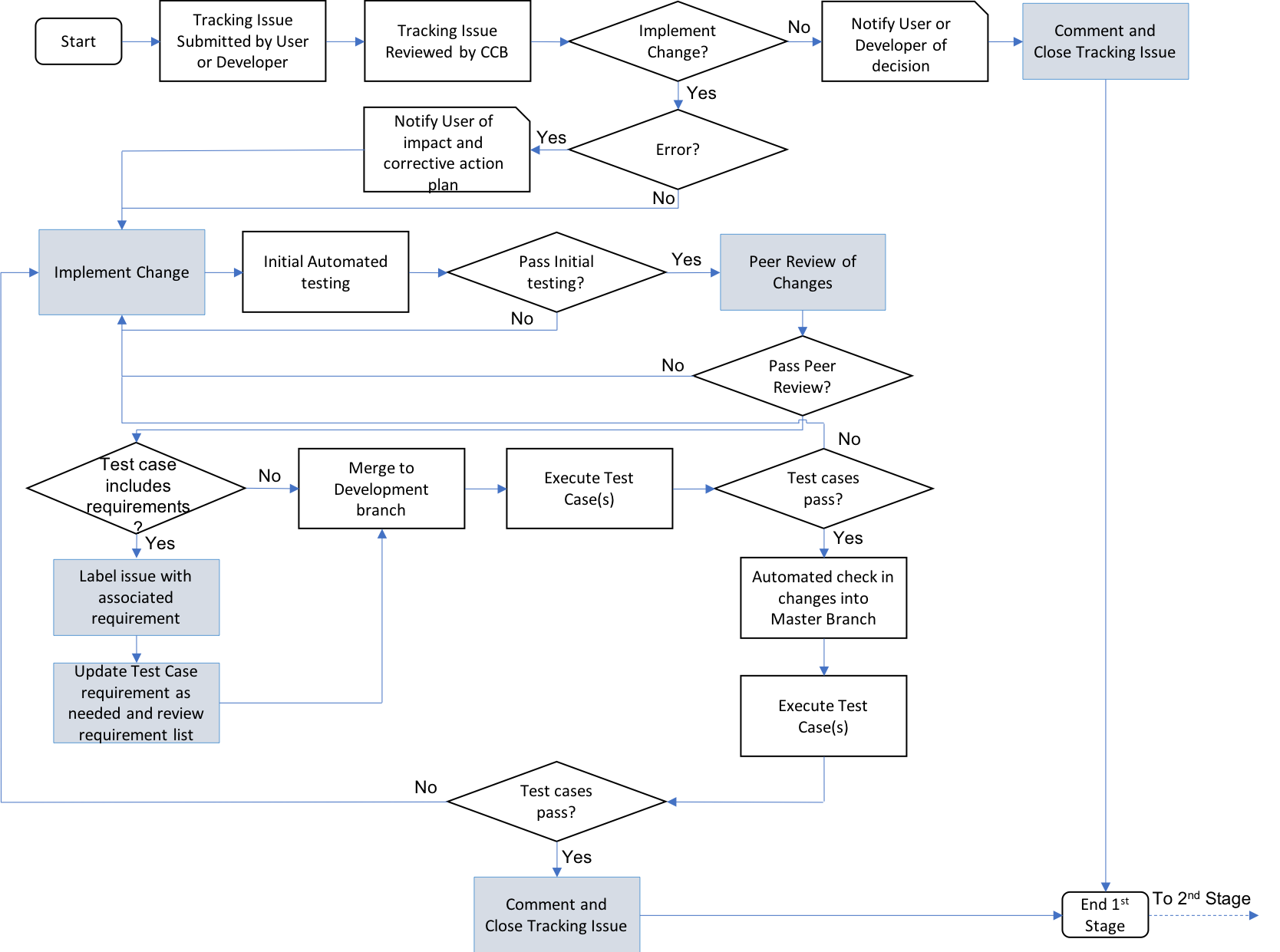


Figure . RAVEN core team’s configuration control process (1st Stage).

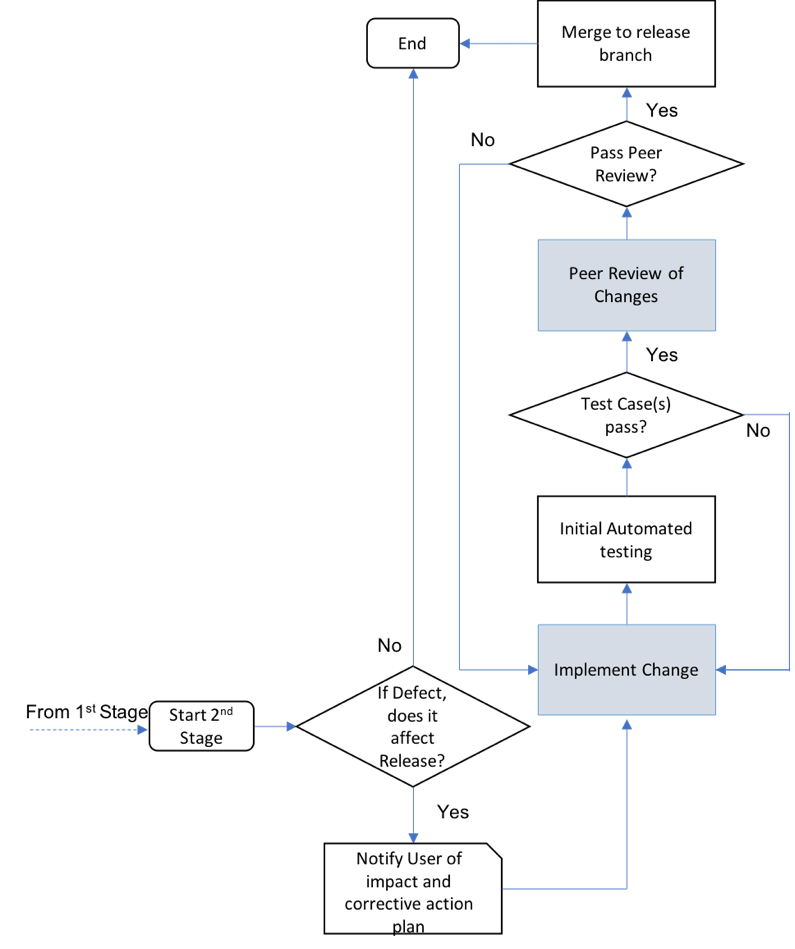


Figure . RAVEN core team’s configuration control process (2nd Stage).

### Requesting Changes

Changes will be initiated for modification of the baseline software, including associated support software, hardware, and/or documentation. For vulnerability patches affecting and including safety software, changes will be tracked, approved by the CCB, and implemented in a timely fashion. *Change requests* (see def., CRs) are submitted in the form of a tracking issue that is created through the issues tracking system found within the GitHub and GitLab software services. CRs may be submitted by development team members based at INL or by external users of the RAVEN software or its supported Plug-ins.

After creation, the CR is pre-screened by a development team member (either RAVEN core or Plug-ins team member) to ensure both a description of the change and rationale for the change are included and are appropriate.

The CR is then classified as either a *task* (GitHub) (see def.) or a *defect* (see def.). An error designation is given if the problem reports RAVEN (or any of its supported Plug-ins) exhibiting abnormal termination, incorrect or unexpected results, or undefined behavior. All developers or users subscribed to the project receive notifications when the issue is opened. At this point in time, the member of RAVEN core team determines if the change should be implemented. If not, the requester of the CR is notified and the tracking issue is appropriately commented and closed.

### Evaluating Changes

The CCB controls and is responsible for the evaluation and disposition of changes for all software, support software, and documentation. The board will consider the impact of the proposed change and assign actions appropriate to the level of impact. If the change is disapproved the decision will be noted on the CR, and the requester will be notified. If additional information is needed, it will be noted and returned to the requester for completion and resubmitted.

There is no established time for a periodic review of the CRs, but based on the judgment of the technical lead an informal schedule will be agreed to so that all CRs are handled in a timely manner. For emergency changes, notification is made to the Asset Owner and/or cognizant technical lead and then implementation of the change is initiated. Processing of the CR may occur following implementation.

The CCB determines the priority and level of rigor of each CR and then evaluates the impact of the change or error on past calculations and how it could affect the present use of the application. The CCB determines the priority based on the following definitions:

* *Safety-related*. Serious problem that affects the accuracy of safety results, past and/or present, and requires immediate attention. The CCB may send additional notifications to affected users for issues marked as both a defect and safety-related.
* *Critical*. CRs necessary to meet critical project deadlines/milestones or prioritized at the discretion of the project manager or technical lead. The CCB may send additional notifications to affected users for issues marked as both a defect and critical.
* *Normal*. Problems affecting the operation/execution of the code, with a low possibility of significantly affecting the results (fine-tuning). Normal priority class items also include problems with method calls, maintenance, modeling problems, user support, and input/output problems.
* *Minor*. Changes to the input/output formats, screen displays, etc., that do not affect the accuracy of the results. Requests for changes to the code, such as enhancements, new development, additional options, making the program more user-friendly, etc.

If the CR is used to report a defect, the CCB will determine whether the use of the application should be suspended while the problem is investigated or until the error is corrected. Users will be notified and provided relevant information including the impact of the error, information on how to avoid the error, corrective action(s) and when the corrective action(s) will be implemented.

The board will also evaluate the impact of the CR on project resources. If possible, the following information will be included on the CR:

* The sequence of events leading up to the suspected problem
* Other unique and/or significant information about the suspected problem that will aid in the evaluation of the problem; for example, limitations and capability differences between versions or anticipated new versions.

Prior to approval and as necessary, the CCB will evaluate impacts to other facility equipment, documentation, and test procedures before approval. After a CR is reviewed, the CCB will determine how to proceed (e.g., initiate scheduling and funding, defer, implementation to a later date, disapprove).

If approved, the changes associated with the CR are then merged into the development branch of the associated repository.

For defects/problems, when closing the issue, CCB member will need to determine if defect/problem applies to the current release branch. In such case, the users need to be notified of the impact and corrective action that will be taken and the CR will be adapted/imported to be merged in the release branch. The control process will then follow the scheme reported in Figure 2.

### Approving or Disapproving Changes

In the RAVEN core team, the CCB can consist of the project manager, technical leads, and development team member(s). Under normal circumstances, the project managers will act as the primary chair of the CCB.

Approvals or disapprovals of the CR are recorded in the associated tracking issue. The tracking issue will contain the name of the person giving final approval and the date of approval.

The CCB may decide to defer approval or disapproval of a CR until a later time. After a decision is made by the board, they will notify the CR requestor of its approval, disapproval, or deferment.

In case of CR that impacts/adds requirements or requirement tests, an additional approval by the CCB chair or its designee (generally the technical lead) is required.

### Implementing Changes: Configuration Status Accounting, Evaluation and Reviews

The Github/Gitlab Pull Request (Merge Request) process is followed for all CRs. This process incorporates the full agile cycle including design, implementation, regression testing, independent design review and approval and integration testing. This section details these stages.

Once the CR has been approved by a CCB member, a requirements review is held to assure the correctness of the proposed modifications stated in the RAVEN software (or supported Plug-in) *change request* (CR, see def.) ticket. In the event that a CR is directly related to new requirements or existing requirement(s), the CR will be labeled with the associated requirement and an additional approval/review by the CCB chair or its designee (generally the technical lead) is required. Requirements reviews will be recorded within the CR.

Appropriate personnel will be assigned to manage and implement the change.

Schedules will be established for each CR activity and for all events affecting the CR implementation. Major CR activities may require more detailed formal scheduling, as well as planning for project funding, manpower, and evaluation of impacts to work activities. Minor CR activities may not require formal written schedules.

The assigned development team member (either RAVEN core or plug-in team member) will implement the requested changes and perform an initial set of automated test suite cases. The automated test suite runs against all the test suite under SQA control to identify impacts to current baselines. Upon successful completion of the automated test suite, the tracking issue is then assigned to the CCB for final approval and further integration into the software and associated build repositories.

The impacted CIs will be baselined and tested, and the appropriate level of regression testing will be performed to ensure that no errors have been introduced into the system. The CCB should consider the level of testing required when evaluating the CR. Once testing is completed, test results will be added as part of completion documentation for the given build and maintained as a record through the Continuous Integration System database.

A member of the CCB will determine if the CR is associated with a requirement-based *test case* (see def.) (annotated in the test case file). If so, the tracking issue will be labeled with the associated requirement and any necessary updates to the requirement associated to the test case will be performed. The requirements traceability matrix can be regenerated at any time to reflect the current state of the repository and test cases. This ensures that all requirements in the SRS are properly associated with the regression test suite.

All V&V task are performed for every code change submitted through the pull request process in GitHub/GitLab. The V&V tasks as outlined in PLN-5554 provide sufficient rigor to meet the INL Quality Assurance program requirements. If the end-use or scope of the information technology (IT) asset changes significantly, the risk analysis must be reviewed. If the results of the analysis change, the M&O manager/project manager shall review the adequacy of the V&V performance to determine if additional V&V activities and/or frequency need to be modified.

The change control process requires a specific set of approvals and successful automated tests for each CI as it is elevated in different branches of the repositories. At several steps during the change commit process, automated tests are executed.

## Subcontractor/Vendor Control

No subcontractors/vendors activities are envisioned for RAVEN and its supported Plug-ins. In case of a new strategy, involving subcontractors, is defined, this plan will be revised.

# Plan Maintenance

The project manager or M&O manager is responsible for maintaining this plan and ensuring that the activities necessary for configuration management are appropriately executed throughout the life cycle of the IT asset.

This plan is controlled per LWP‑1201, “Document Management.” Revisions to this plan will occur on an as-needed basis as a result of reviews, audits, and requested changes. Modifications to this plan must be independently reviewed approved by the Asset Owner.