Laboratory-wide	Template		eCR Number:	619940

Document ID: PLN-5552
Revision ID: 0
Effective Date: 01/30/2019

Software Quality Assurance Plan (SQAP)

RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance.

RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan

PLN-5552

Prepared by:

Andrea Alfonsi
IT Project/M&O Manager — Date

Reviewed by:

See eCR #
Independent Reviewer — Date

Approved by:

See eCR #
Asset Owner — Date

CONTENTS

1. PURPOSE 5

1.1 RAVEN Description 5

1.2 Software Lifecycle 6

1.3 Assumption and Constraints 6

1.4 Deviation Policy 6

2. REFERENCES 7

3. DEFINITIONS AND ACRONYMS 8

3.1 Definitions 8

3.2 Acronyms 13

4. MANAGEMENT 15

4.1 Organization 15

4.2 Roles and Responsibilities 15

4.3 Tasks 19

4.4 Applicable Policies, Directives, and Procedures 20

5. CONFIGURATION MANAGEMENT 20

5.1 Configuration Identification 20

5.1.1 Identifying Configuration Items 20
5.1.2 Naming Configuration Items 21
5.1.3 Acquiring Configuration Items 21
5.2 Configuration Control 21

5.3 Requesting Changes 24

5.4 Evaluating Changes 25

5.5 Implementing Changes: Configuration Status Accounting, Evaluation and Reviews 27

6. SUBCONTRACTOR.VENDOR 28

7. DOCUMENTATION 28

7.1 Minimum Documentation Requirements 28

7.2 Other Documentation 29

8. STANDARDS, PRACTICES, CONVENTIONS, AND METRICS 29

8.1 Content 29

8.1.1 Software Coding Standards 30
8.1.2 Commentary Standards 30
8.1.3 Testing Standards and Practices 30

9. SOFTWARE REVIEWS 31

9.1 Minimum Requirements 31

9.1.1 Requirements Reviews 31
9.1.2 Design Reviews 31
9.1.3 Acceptance Review 31
9.1.4 Change Request Approval Check List 32

10. TESTING 32

10.1 V&V Overview 32

10.1.1 Test & V&V Objectives 32
10.1.2 Master Schedule 34
10.1.3 Specific meaning of V&V activities for RAVEN software 34

10.2 TYPES OF TESTS TO BE EXECUTED 34

10.3 Test Automation 36

10.4 APPROVAL REQUIREMENTS 36

10.5 Requirement tests 37

10.6 Other tests 37

10.7 TEST DEFINITION TASKS AND RESPONSIBILITIES 37

11. V&V PROCESSES 38

11.1 V&V Reporting Requirements 41

11.2 V&V Administrative Requirements 42

11.2.1 Anomaly Resolution and Reporting 42

11.3 Task Iteration Policy 43

11.4 Control Procedures 43

12. PROBLEM REPORTING AND CORRECTIVE ACTION 44

13. TOOLS, TECHNIQUES, AND METHODOLOGIES 44

14. SUPPLIER CONTROL 46

15. RECORDS COLLECTION, MAINTENANCE, AND RETENTION 46

16. TRAINING 46

17. RISK MANAGEMENT 47

17.1 Safety Software Determination 47

17.2 Quality Level Determination 47

18. ASSET MAINTENANCE 47

18.1 Business Requirements 47

18.2 Schedule and Budget and Summary 48

18.3 Evolution of the Plan 48

18.4 System Hardware and Operating Systems 48

18.5 Backup and Recovery 48

19. MAINTENANCE AND OPERATIONS PLANNING 49

19.1 M&O Initiation 49

19.1.1 Estimation Plan 49

19.1.2 Staffing Plan 49

19.1.3 Training Plan 49

20. M&O Work Plans 50

20.1 Work Activities 50

20.2 Resource Allocation 50

20.3 Budget Allocation 50

20.4 Acquisition Plan 51

21. M&O ASSESSMENT AND CONTROL 51

21.1 Requirements and Design Control Plan 51

21.2 Subcontractor Management Plan 52

22. SUPPORTING PROCESS PLANS 52

22.1 Communication and Publicity 52

22.2 Assessments 52

22.3 Retirement 52

1. PURPOSE

Software quality assurance (SQA) is a set of activities necessary to provide adequate confidence that a software item or product conforms to the set of functional and technical requirements specified for that item. This plan presents the required activities to enable consistent SQA implementation within the Risk Analysis and Virtual ENvironment (RAVEN) Software and any RAVEN plug-ins (see def.). It provides a standardized method of capturing software requirements, how those requirements will be implemented, how the software will be tested, how changes to the software will be controlled, and how software deficiencies will be handled. This Software Quality Assurance Plan (SQAP) establishes the software Quality Assurance program for RAVEN. It covers the periods of software development, maintenance and operations (M&O), and retirement. It implements applicable requirements in conformance with PDD-13610, "Software Quality Assurance".

1.1 RAVEN Description

RAVEN is a flexible and multi-purpose uncertainty quantification (UQ), regression analysis, probabilistic risk assessment (PRA), data analysis and model optimization software. RAVEN was designed and has been developed to provide the capabilities needed to perform:

- Uncertainty Quantification,
- Sensitivity Analysis / Regression Analysis,
- Probabilistic Risk and Reliability Analysis (PRA),
- Data Mining Analysis
- Model Optimization

RAVEN is operational within multiple projects. Ongoing support of RAVEN is required for the purpose of adding functionality, correcting computational errors and improving the performance of the RAVEN software.

- RAVEN is maintained by a team of scientists/researchers, referred to herein as the RAVEN core team (see def.). RAVEN maintenance and operations, performed by the RAVEN core team, is an ongoing activity.

- This plan covers the maintenance of all existing and future components of RAVEN. This includes, but is not limited to, servers, server software, user workstations, RAVEN software, and control documents. Changes to this document will be completed through the Electronic Change Request (eCR) process.

1.2 Software Lifecycle

RAVEN is using an Agile life cycle methodology. The life cycle will be performed in an iterative manner and address the requirements, design, implementation, testing, installation and checkout, operations and maintenance, and retirement phases.

1.3 Assumption and Constraints

- The RAVEN core 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.
- 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.
- The hardware that serves RAVEN is managed by the High-Performance Computing Group. The hardware is considered a configuration item (see def.) for the RAVEN asset, and changes impacting the RAVEN framework must be reviewed by the RAVEN technical lead or designee; however, the management of the hardware is outside the scope of this plan.
-

1.4 Deviation Policy

All deviations from this plan require management approval. Whether planned or unplanned, if any deviation from this plan is necessary, the following components will be determined:

- Identification of task affected.
- Reasons for deviation defined.
- Effects on the quality of the project.
- Time and resource constraints affected.

A deviation report will be generated, and authorization will be required. Deviations that violate requirements must be documented within the relevant issue.

2. REFERENCES

The following source documents apply to this SQAP:

- 29 USC 794d, Section 508 Workforce Investment Act of 1998
- INL/EXT-18-44465, "RAVEN User Documentation"
- ISO/IEC/IEEE 24765:2010(E), "Systems and software engineering — Vocabulary"
- PDD-13610, "Software Quality Assurance Program."
- PDD-13000, "Quality Assurance Program Description"
- LWP-1201, "Document Management"
- LWP-1202, "Records Management"
- LWP-1305, "Acquisition of Computer Hardware/Software Resources"
- LWP-1306, "Management of IT Asset Minimum Security Configurations," Rev. 1, December 23, 2013.
- LWP-1401, "Preparing and Releasing Scientific & Technical Information Products"
- LWP-4001, "Material Acquisitions"
- LWP-4002, "Service Acquisitions"
- PLN-4653, "INL Records Management Plan"
- SDD-513, "RAVEN Software Design Description (SDD)"
- SPC-2366, "RAVEN Software Requirements Specification (SRS) and Traceability Matrix"

3. DEFINITIONS AND ACRONYMS

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

3.1 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. 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.

Agile development. Agile development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). It prescribes adaptive planning, continuous development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.

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 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]

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.

Commercial off-the-shelf. (COTS) Usually refers to software purchased from a vendor "as-is" with minimal customization or configuration options that meets a requirement.

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 (CI). 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]).

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.

Continuous Integration System (CIS). A system, linked to a central version control repository, such as *GitHub* and *GitLab* (see def.), aimed to automatically build and test a targeted software. Examples are CIVET, Jenkins, and GitLab Continuous Integration.

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, fault or failure 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, C++, Fortran and Python 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 many applications/extensions/Plug-ins built/developed on the RAVEN software. The

CIS (see def.) 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/developer.

Major Change. A revision to software that, in the best judgment of authorizing personnel, 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 address program errors and problems. Pull requests allow informing others about changes pushed to a repository on a *version control system (see def.).* 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, as well as integrate changes into the maintained code.

Quality grade. The grade applied to the level of quality activities to be applied to the specific task or activity. Current quality grades are Nuclear Use QL and Commercial Use Quality Levels (QLs) High, Medium, and Low.

RAVEN core team. INL personnel who are in charge of the development of the RAVEN framework or software applications/extensions/plugins that are based on the RAVEN framework. A list of the current components of the RAVEN core

team can be found at https://github.com/idaholab/raven/wiki/AboutUs#raven-core-team

RAVEN Software. Open source software that resides in a public repository (GitHub) that provides the capabilities needed to perform Uncertainty Quantification, Probabilistic Risk Assessment, Data Analysis, Validation and Parameter Optimization.

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.

Retirement. Permanent removal of an asset (e.g., system or component) and associated support from its operational environment.
[ISO/IEC/IEEE Std 24765-2010 edited]

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).

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, 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 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. (2) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.

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]. As described in SDD-513, "RAVEN Software Design Description," RAVEN does not own Physical models and the Validation is performed verifying the algorithms/methods with analytical solutions (if applicable).

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]

Version Control System. It is the system aimed to support the management of changes to files, in general, and computer programs, in particular. Changes are usually identified by a number, letter code or unique alphanumeric identifiers, termed the "revision number", "revision level", or simply "revision". Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged. Examples of Version Control Systems are GitHub and GitLab (see def.)

3.2 Acronyms

ASME American Society of Mechanical Engineers

BEA	Battelle Energy Alliance
CCB	Change Control Board
CFR	Code of Federal Regulations
CI	Configuration Item
CIS	Continuous Integration System
CM	Configuration Management
CMP	Configuration Management Plan
COTS	Commercial off-the-shelf software
CR	Change Request
CSV	Comma Separated Value
DOE	Department of Energy
EA	Enterprise Architecture
EDMS	Electronic Document Management System
IAS	Integrated Assessment System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INL	Idaho National Laboratory
ISMS	Integrated Safety Management System
ISO	International Organization for Standardization
IT	Information Technology
LST	List
LWP	Lab-wide Procedure
M&O	Maintenance and Operations
NQA	Nuclear Quality Assurance

POSIX	Portable Operating System Interface
PRA	Probabilistic Risk Assessment
QA	Quality Assurance
QL	Quality Level
QLD	Quality Level Determination
RTM	Requirement Traceability Matrix
RAVEN	Risk Analysis and Virtual ENvironment
SRS	Software Requirements Specification
SSD	Safety Software Determination
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
USGCB	U.S. Government Configuration Baseline
V&V	Verification and Validation

4. MANAGEMENT

4.1 Organization

The RAVEN core team is responsible for the project activities of the RAVEN software (see def.).

4.2 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. Table 1 identifies roles and responsibilities.

Table 1. Roles and responsibilities.

Role	Responsibilities
Management	• Provide funding and staffing for RAVEN and RAVEN plugins/extension software activities

Role	Responsibilities
	• Assign personnel and ensure they are properly qualified and trained to perform SQA tasks. Refer to Training Section of this plan for further detail. • Ensure corrective actions are implemented as needed
Asset Owner	• Acquire and dedicate IT materials and services in accordance with INL acquisition policy and this plan. • Responsible for administration and execution of this plan. • Ensure the completion of the QLD as part of a criticality/risk analysis. • Participate as necessary on the change control board (see def., CCB) as needed and act as final authority when necessary. • Act as the final authority for the approval/disapproval of change requests (see def.). • Review acceptance test and approve asset for deployment.

Role	Responsibilities
Project Manager	• Interface for all the internal and external RAVEN contacts/customers • Create/update asset portfolio including total expected life-cycle cost information. Review and approve management plan documentation. • Create/revise management plan documentation. Acquire and dedicate IT materials and services in accordance with INL acquisition policy. • Manage and resolve problems per this plan. Create/revise management plan documentation. • Provide status reports to management per communication plan. Coordinate independent review of management plan, requirements, and design documentation. • Act as the chair of the CCB. • Approve/disapprove and status all change requests.

Role	Responsibilities
Technical Lead	• Document test procedures and instructions for use. • Coordinate execution of implementation review. • Maintain requirements/design baseline (see def.). • Final approval on design reviews. • Oversight of design implementation and integration testing activities when applicable. • Assign system administrator duties as needed. • Resource Allocation dispatching in conjunction with the Project Manager • Conduct requirements, design, and implementation reviews. • Approve testing results for release of a new software version. • Identify and manage configuration items (see def.). • Ensure implementation and verification (see def.) of change and document as required by this plan. • Participate on the CCB. • Place assets under version control. • Establish baseline (see def.) of the asset prior to acceptance test. • Evaluate issues (see def.) and anomalies. • Initiate component and integration tests prior to system test. • Ensure implementation and verification activities are complete and document as required. • Coordinate execution of configuration audits.
Independent Reviewer	• Review management plan and participate in requirements, design, and implementation reviews. • Provide status reports to management per communication plan.
Change Control Board (CCB)	• Review and approve change requests. • Evaluate test results as part of the approved changes.

Role	Responsibilities
Software Developer	• Perform design, implementation and testing of the software code. • Adhere to this plan.

4.3 Tasks

Table , Software quality assurance tasks, identifies the software quality assurance tasks to be performed. The "schedule" column identifies when the tasks are performed, and the entrance and exit criteria for each stage are also established.

Table 2. Software quality assurance tasks.

Task	Schedule	Entry Criteria	Exit Criteria
Risk Analysis	Per major release	Business requirements	Approved safety software determination (SSD) and quality level determination (QLD)
Management Plan review and approval	As needed	Draft management plan	Approved by Asset Owner.
Requirements review and approval	Per major release	Draft design description	Approved by Asset Owner.
Design review and approval	Per major release	Draft design description.	Approved by Asset Owner.
Implementation review	Per release	Baselined software prior to system test.	Documented code walk-through to ensure consistency of software and supporting documentation including traceability of requirements through the life cycle.
System Test	As required by SCMP.	Completed implementation review.	Approved system test by test case personnel.
Acceptance Test	Per release	Completed system test.	Asset approved by Asset Owner or Technical Leader.
Problem Resolution	As needed	Problem report submitted.	Closed problem report.

4.4 Applicable Policies, Directives, and Procedures

The INL Software Quality Assurance (SQA) Program is implemented in compliance with PDD-13610, "Software Quality Assurance". 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. The test cases/procedures are maintained in RAVEN GitHub and/or GitLab (Plug-ins) repository and accessible by any individual/user.

5. CONFIGURATION MANAGEMENT

The purpose of this section is to document the configuration management (CM) activities, plan management, and maintenance needed to assure proper configuration of the RAVEN software and its supported Plug-ins. Specifically, it outlines the configuration identification, controls, status accounting, evaluation, and reviews.

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.

5.1 Configuration Identification

5.1.1 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. A software baseline that includes the CIs will be established and maintained throughout the software life cycle.

For CIs, 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 core and RAVEN supported Plug-ins teams' technical leads to determine the level of detail for the CI list.

5.1.2 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."

5.1.3 Acquiring Configuration Items

The Asset owner(s), 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.

5.2 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 supported Plug-ins (covered by scope of this plan).

RAVEN software and supported Plug-ins operate under an agile 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 (USGCB MSC), follow the Computer Security and Operational Variance 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 1 and Figure 2 for a depiction of the RAVEN core team's configuration control process (with peer-review included).

Configuration baselines are established for each revision.

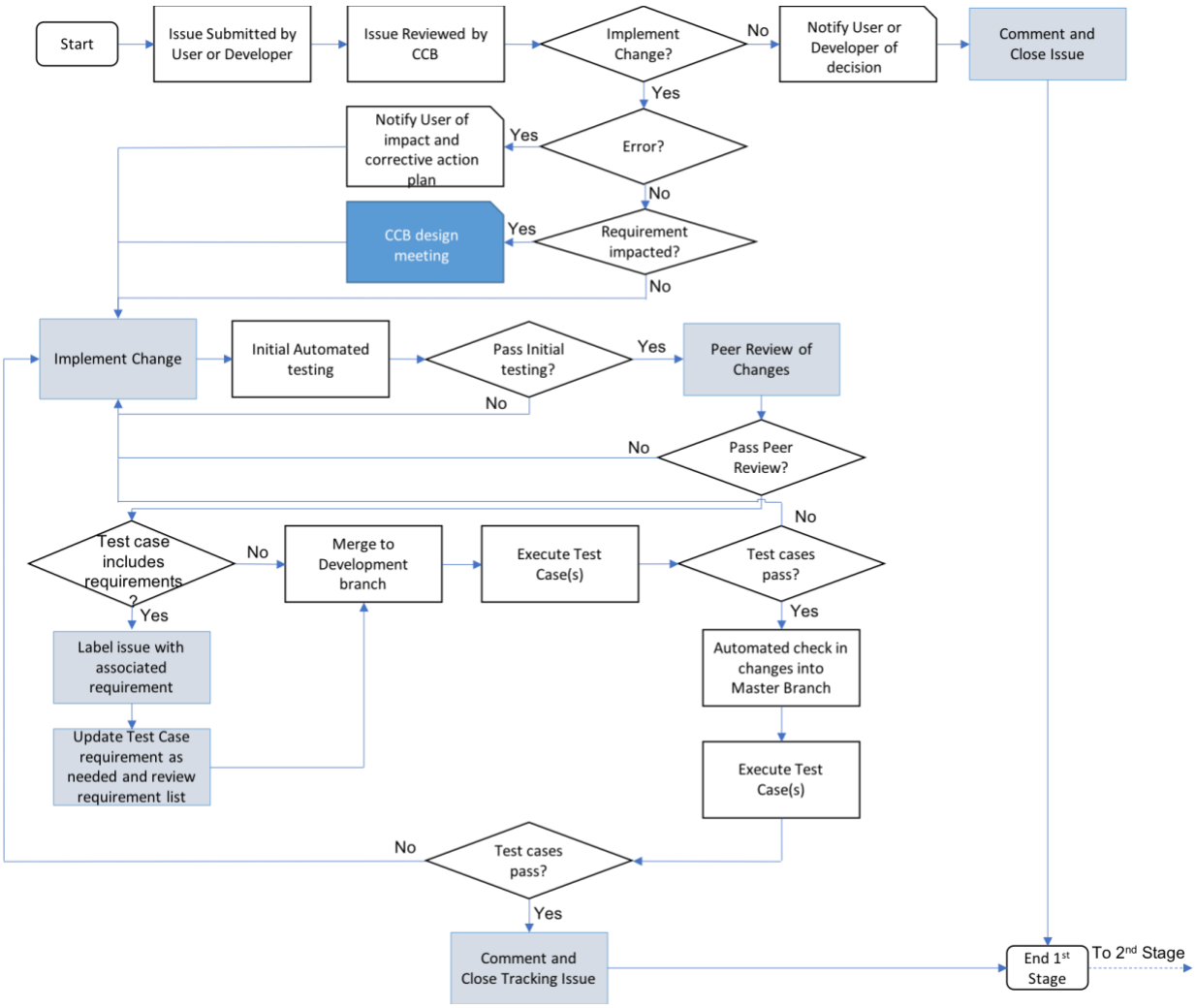

Figure 1. RAVEN core team's configuration control process (1st Stage).

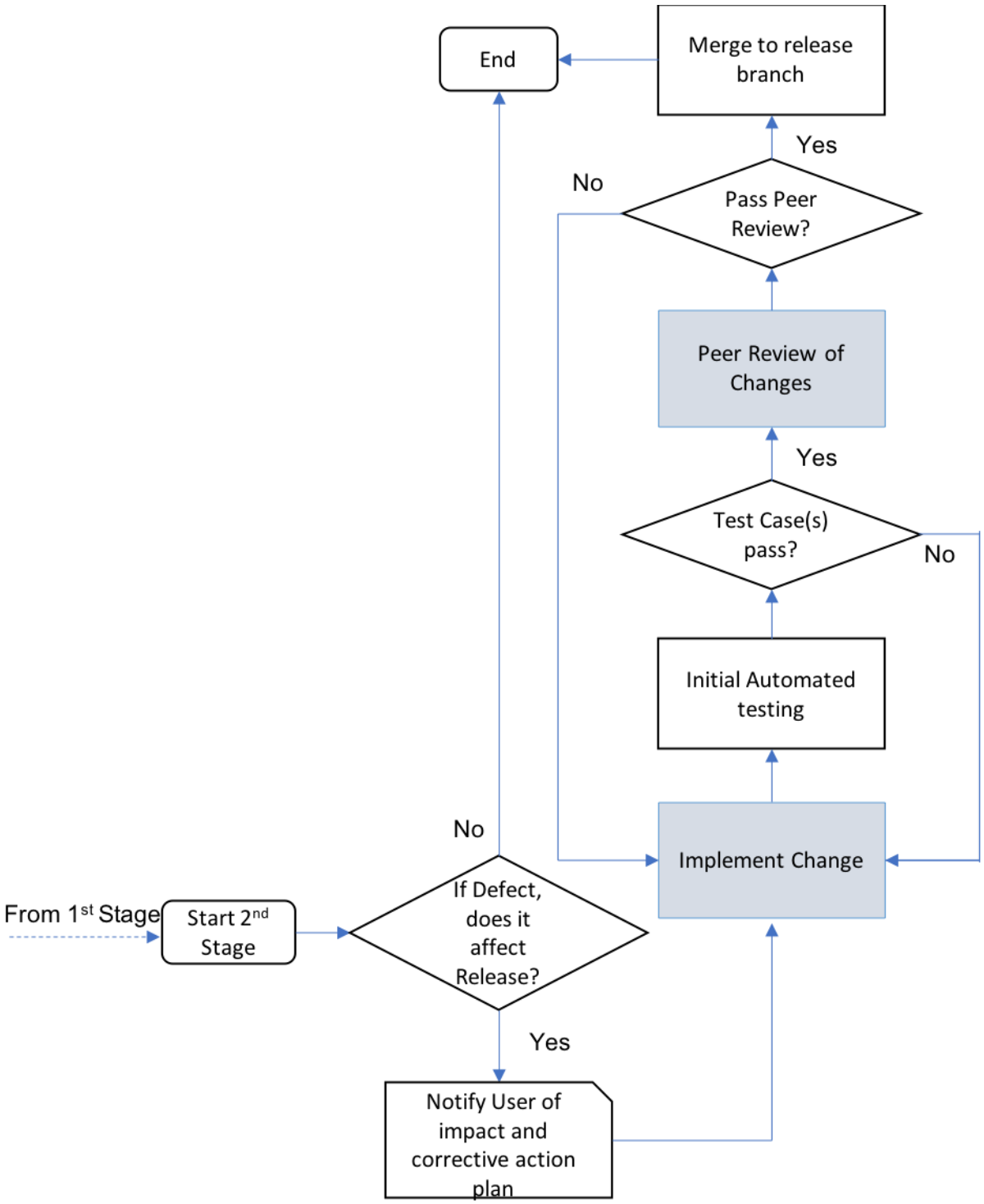

Figure 2. RAVEN core team's configuration control process (2nd Stage).

5.3 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 request (CRs)* (see def.) are submitted in the form of a tracking issue that is created through the issues tracking system found within the *Version Control (see def.) System* (e.g. *GitHub/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 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.

5.4 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. 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 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 task or defect on past calculations and how it could affect the present use of the application. The CCB determines the priority based on the following definitions:

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 determines if defect/problem applies to the current release branch. In such case, the users are notified of the impact and corrective action that will be taken and the CR is 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.

5.5 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 is 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 are assigned to manage and implement the change.

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

The assigned development team member (either RAVEN core or RAVEN supported 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 are then baselined and tested, and the appropriate level of regression testing 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 are 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 then determines if the CR is associated with a requirement-based test case (see def.) (annotated in the test case file). If so, the tracking issue is labeled with the associated requirement and any necessary updates to the requirement associated to the test case is 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 SPC-2366, "RAVEN Software Requirements Specification," are properly associated with the regression test suite.

All verification (see def.) & validation (see def.) (V&V) task are performed for every code change submitted through the pull request process in GitHub/GitLab. If the end-use or scope of the asset changes significantly, the risk analysis must be reviewed.

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.

6. SUBCONTRACTOR.VENDOR

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.

7. DOCUMENTATION

The purpose of this section is to define the minimum documentation required to properly implement the SQA requirements. At all times during the life cycle of RAVEN, the following documents will be maintained as part of the Asset Portfolio.

7.1 Minimum Documentation Requirements

As a minimum, the following documentation is required for the RAVEN software and supported RAVEN supported Plug-ins. These documents are managed as

records in accordance with Section 15, "RECORDS COLLECTION, MAINTENANCE, AND RETENTION."

The following documentation is required as a minimum:

Document	Record Location	ID
Software Quality Assurance Plan	Electronic Document Management System (EDMS)	PLN-5552
Software Test Plan and Verification & Validation	GitHub/GitLab	PLN-5552
Software Requirements Specification and Traceability Matrix	GitHub/GitLab	SPC-2366
Software Design Description	GitHub/GitLab	SDD-513
User Documentation (see def.)	GitHub/GitLab	INL/EXT-15-34123, INL/EXT-18-44465, INL/EXT-16-38178

7.2 Other Documentation

In addition to the above documents, the following are created during the procurement and baselining of the project. These may be used in support of Change Control Request implementation and M&O activities.

- SSD-000649, "Risk Analysis and Virtual Environment"
- QLD, "RAVEN Quality Level Determination"
- RAVEN Enterprise Architecture Entry #331229

All documents will be managed according to LWP-1201, "Document Management."

All records generated as part of this plan will be processed and managed according to LWP-1202, "Records Management."

8. STANDARDS, PRACTICES, CONVENTIONS, AND METRICS

8.1 Content

The standards for RAVEN and RAVEN supported Plug-ins are maintained/recorded in the RAVEN GitHub repository (Wiki section). Any developer of the RAVEN software or RAVEN supported Plug-ins need to be aware of the standards and to follow the development guidelines.

The RAVEN standards evolve around the following macro-areas:

- Software Coding Standards
- Commentary Standards
- Testing Standards and Practices

8.1.1 Software Coding Standards

The RAVEN software imposes a coding standard on all source code within the repository. This standard is publicly maintained on the RAVEN GitHub repository wiki website (https://github.com/idaholab/raven/wiki/RAVEN-Software-Coding-Standard) and enforced through the continuous integration testing system.

8.1.2 Commentary Standards

The RAVEN software imposes a commentary standard on all source code within the repository. The standard is aimed to fully describe any module/method in the source code, guaranteeing the automatic generation of software documentation via doxygen (see def.). This standard is publicly maintained on the RAVEN GitHub repository wiki website (https://github.com/idaholab/raven/wiki/RAVEN-Software-Commentary-Standard) and enforced through the continuous integration testing system.

8.1.3 Testing Standards and Practices

The RAVEN software imposes a testing standard and practices on all the capabilities/methods of the RAVEN software. This standard is publicly maintained on the RAVEN GitHub repository wiki website (https://github.com/idaholab/raven/wiki/RAVEN-Testing-Standards-and-Practices) and enforced through the review process by a member of the CCB.

9. SOFTWARE REVIEWS

Software reviews serve to assure appropriate progression to the next phase in the software lifecycle. Software reviews provide an independent perspective from those directly involved in the software development process. The independent reviewer will participate in the process of these reviews on a risk-based graded approach based on the RAVEN subject matter expert (SME) assessment. Any CCB member is considered independent reviewer for the documentation associated with the CR.

9.1 Minimum Requirements

At a minimum, the following reviews will be conducted. All comments/change histories are retained in the *GitHub* system. Any design aspects that are not resolved at the time of the review are addressed in follow-up *issues* (GitHub) (see def.).

At a minimum, the following reviews will be conducted (if applicable):

9.1.1 Requirements Reviews

In case of CR that modifies and/or add new requirements for the RAVEN software, the CCB chair or designee shall perform a requirement review during the initial stages of planning, before procurement and configuration of the asset. The results will be documented in SPC-2366, "RAVEN Requirements Traceability Matrix." An additional record of the review will be retained in GitHub (see def.).

9.1.2 Design Reviews

One or more independent design reviews are required for all design changes to evaluate the technical adequacy of the design approach and ensure internal completeness, consistency, clarity, and correctness of the software design. In addition, it is required to demonstrate that software design is traceable to the software requirements. These reviews will include review of test results and be recorded, with identification of the reviewer, within the CR. This is implemented as a series of comments and date fields used by the RAVEN core team (being part of the CCB) to record the required reviews and approvals prior to acceptance for use. Any changes to source code, comments, or documentation within the code repository trigger automated testing.

9.1.3 Acceptance Review

Review is performed by the Software Technical Leader or Project Manager to ensure compliance with the approved software requirements.

Automated *regression testing* (see def.) system results are considered to be part of the acceptance review.

9.1.4 Change Request Approval Check List

In order to guide the review process of any CR in the RAVEN software, a PR (see def.) check list needs to be satisfied. If any of the required checks are not satisfied the CR developer and the reviewer(s) need to document the reason why a certain check is not applicable. The PR check list is publicly maintained on the RAVEN GitHub repository wiki website (https://github.com/idaholab/raven/wiki/development-checklists#peer-review-checklist-for-merge-requests).

10. TESTING

The goal of software *validation* (see def.) is to confirm that the requirements for a specific intended end use have been fulfilled. Software *verification* (see def.) evaluates a system or component to confirm that specified conditions have been satisfied and provides formal proof of correctness.

10.1 V&V Overview

10.1.1 Test & V&V Objectives

Test procedures or plans will specify the following as applicable:

- required tests and test sequence
- required ranges of input parameters
- identification of the stages at which testing is required
- criteria for establishing test cases
- requirements for testing logic branches
- requirements for hardware integration
- anticipated output values
- acceptance criteria
- reports, records, standard formatting, and conventions
- performance testing

Any developer, including externals, are responsible for ensuring the creation of a *test case* (see def.) that covers the new capability or code change. The *CCB* (any of its member not directly involved in the *CR*) is responsible, through the help of the Review Check Lists (see def.), for verifying that an appropriate test case is provided, and passes based on the supplied acceptance criteria. This verification is performed for any CR and failing to meet these requirements shall conclude in rejecting the *CR* by the *CCB* member/reviewer. The process for handling *CRs* that modify or add requirements is discussed in Section 5, Configuration Management Activities.

RAVEN is *open source* (see def.) software that is maintained and stored in *GitHub* (see def.), a public repository. In order to align the testing and V&V activities of the software with the nature of the *Agile development process* (see def.), the verification of the software has been designed in a multi-stage automated testing suite, using the *Continuous Integration System* (CIS) (see def.) both in GitHub, for open source software, and in *GitLab (*see def.), for protected software (e.g. RAVEN Plug-ins).

The main scope of the automated testing is to guarantee that any capability is properly tested and that new addition to the software do not impact the functionalities of the already-deployed capabilities.

Four types of testing, unit, integration, system, and deployment, are covered by the RAVEN framework and, optionally, each RAVEN Plug-in.

The project manager/technical leader oversees the testing and verification and validation (V&V) activities, including the analysis of test coverage and the determination of when new tests are necessary. The test coverage analysis is performed during the code review activities conducted by the *RAVEN core team* (see def.), and it is determined at that step in the process if one or more new tests needs to be created. V&V activities are distributed among the RAVEN core team.

Every time a new development or capability is performed by a software developer, the following shall be determined:

- Required test activities and method of documentation (e.g., test plans, procedures, checklists, etc.);
- Required *support software* (see def.) (e.g., automated test scripts, fault insertion tools, etc.);
- Type and extent of required testing; and
- Required reviews and approvals.

A component (or more) of the *change control board* (*CCB)* (see def.), not being part of the development, shall review the correct documentation of the tests and ensure that the documentation includes approved requirements (when necessary) that have valid acceptance criteria. This documentation may include:

- Documentation of the tests including acceptance criteria. The documentation procedure is defined in the RAVEN wiki page (https://github.com/idaholab/raven/wiki/Developer_Information#developing-regression-tests)
- Software Requirements Specification or equivalent requirements document;
- Requirements Traceability Matrix;
- Software Design Description for guidance on testing methodologies and the operating environment (i.e., software, firmware, and hardware elements) to be used during testing;
- *User documentation* (see def.)

The CIS will verify that the provided documentation ensures that the software demonstrates adherence to the documented requirements and that the software produces correct results.

10.1.2 Master Schedule

The V&V tasks (as captured in the automated tests) are executed automatically for every change to RAVEN software (i.e. source code). At several steps during the change commit process, automated tests are executed.

10.1.3 Specific meaning of V&V activities for RAVEN software

RAVEN is a multi-purpose uncertainty quantification (UQ), probabilistic risk assessment (PRA), Parameter Optimization and Data Analysis software; RAVEN does not own any physical model (i.e. it does not model/simulate any physical phenomena or system). Consequentially, the validation (see def.) of the RAVEN software is mostly related to the verification (see def.) of the models/capabilities with analytical testing or process (when applicable).

10.2 TYPES OF TESTS TO BE EXECUTED

Tests are defined using an input file syntax, which specifies what the test should do, the inputs, and the post conditions for determining test success or failure; and assuring that the software produces correct results. The guidelines for the creation of a new test are reported in the RAVEN wiki page (https://github.com/idaholab/raven/wiki/Developing-Regression-Tests). Any test case that is connected with a requirement or modify/add a new requirement shall be tagged with the associated requirement ID.

Acceptance Criteria for each test is defined by the Test type (defined below).

The collection of Test types ensure that the software properly handles abnormal conditions and events as well as credible failures, does not perform adverse unintended functions, and does not degrade the system either by itself, or in combination with other functions or configuration items.

The Test types and acceptance criteria for each are as follows:

- CSVdiff: A test case that runs a simulation, terminates without error, and produces a previously defined comma separated value solution within a predefined tolerance (usually to at least single precision accuracy or better). The order of data in the CSV must exactly match the reference solution file.

- UnorderedCSVDiffer: A test case that runs a simulation, terminates without error, and produces a previously defined comma separated value solution within a predefined tolerance (usually to at least single precision accuracy or better). The order of data (rows) in the CSV can be different with respect the previously defined file. *Note:* This Test is generally used when multiple parallel executions of an underneath model are performed, and the collection of the data can be unsynchronized depending on the latency of the network/machine. ***This test is only allowed if a parallel test is created.***

- TextDiff: A test case that runs a simulation, terminates without error, and produces a previously defined text file that matches a reference solution file.

- XMLDiff: A test case that runs a simulation, terminates without error, and produces a previously defined Extensible Markup Language (XML) solution within a predefined tolerance (usually to at least single precision accuracy or better).

- RAVENImageDiff: A test case that runs a simulation, terminates without error, and produces a previously defined image or picture within a predefined tolerance (in terms of pixel difference).

- RavenErrors: A test case that runs and produces a specified console output or output pattern and terminates with an expected error code or message.

- HPCinteraction: A test case that runs a simulation in a High-Performance Computing System using its native Job Scheduler and Workload manager (e.g. Portable Batch System – PBS), terminates without error.

Any of the above described tests can be performed both in system/integral test configuration (RavenFramework) or unit-testing (RavenPython).

In addition to the above reported Test types, for any CR the following tests are performed:

- Documentation Test: The CIS tests that the User Documentation and SQA Documentation can correctly be generated.

- XSD Schema Validation: The CIS tests that all the test files are compliant with the prescribed input syntax.

- Code Standard Validation: The CIS tests that all the source code is compliant with the RAVEN software coding standards (e.g. source code syntax, formats, documentation, etc.).

- Code Coverage: The CIS tests that at least the 80% of the source code is tested by the test suite.

10.3 Test Automation

Testing is performed automatically as part of the CIS process when a user commits a change to the repository. The automated tests that are executed at subsequent steps in the process vary in scope and type and are described in Table 2. Tests of the framework across multiple platforms (operative systems and versions) are executed with each *pull request* (see def.).

In order to pass acceptance testing, all test cases are expected to pass under the environments identified in the configuration items for RAVEN software.

Use of the automated tests is integrated directly into GitHub and GitLab for RAVEN supported (and not-open source) Plug-ins, and as such does not require additional training other than general familiarity with performing a pull request in GitHub (or GitLab).

Results from each test execution are maintained in the CIS database, in an approved records repository along with results from the timing executions and code coverage.

10.4 APPROVAL REQUIREMENTS

The RAVEN and RAVEN supported Plug-ins rely on a heavy automation of the verification and testing of any new or modified capability. This approach is required for the nature of the *Agile development* process. As mentioned in the previous section, any CR in the source code needs to be accompanied with a new (or modified) test to assess the correctness of the code and its functionality.

Depending of the type of test case that is added or modified, two different approval processes are followed:

10.5 Requirement tests

This category is about to test any functionality that is linked to any new or assessed requirements.

Table 3 - Requirement tests' responsibilities.

Test Case Reviewer(s):	Chair of the *CCB*, Technical Leader and Independent Reviewer (Member of the *CCB*)
Test Result Reviewer and Approver:	Chair of the *CCB* or Technical Leader and Independent Reviewer (Member of the *CCB*)
Acceptance Test Case Reviewer(s):	Chair of the *CCB*, Technical Leader and Independent Reviewer (Member of the *CCB*)
Acceptance Result Reviewer(s):	Automated CIS
Acceptance Result Approver:	Automated CIS

10.6 Other tests

This category is about to test any functionality that is not linked to any specific requirement (e.g. infrastructure tests, verification tests, etc.).

Table 4 - Other tests' responsibilities

Test Case Reviewer(s):	Independent Reviewer (Member of the *CCB*)
Test Result Reviewer and Approver:	Independent Reviewer (Member of the *CCB*)
Acceptance Test Case Reviewer(s):	Independent Reviewer (Member of the *CCB*)
Acceptance Result Reviewer(s):	Automated CIS
Acceptance Result Approver:	Automated CIS

10.7 TEST DEFINITION TASKS AND RESPONSIBILITIES

This section summarizes the tasks and associated roles in the definition of the test cases and their approval.

Table 5 - Tasks and responsibilities for tests creation.

Tasks	Responsibility
1. Complete programming and test creation	Developer of the proposed CR
2. Test data creation	Developer of the proposed CR
3. Set up test environment	Automated via CIS
4. Migrate services to test environment	Automated via CIS
5. Set up test database	Automated via CIS
6. Prepare test cases	Developer of the *CR*
7. Conduct test, record results, and communicate to the developers	Automated via CIS
8. Make corrections and updates to the processes	Developer of the *CR*
9. Review and approve final results of the test	Independent reviewer part of the *CCB* and Technical Leader (or Chair of *CCB*) in case of requirement test.

Note: *The above steps need to be conducted for every type of testing*

11. V&V PROCESSES

The V&V tasks, both automatic and manual, occur for the RAVEN software and RAVEN supported Plug-ins as part of the act of committing a CR (performing a "pull request"). These tasks are performed by the RAVEN core team during the development and testing phases of the *software life cycle* (see def.). These tasks are listed in the Table below.

Table 6 – Verification and Validation tasks.

Task	Who	What

"Pull Request" Testing	Automated	- This testing is automatically executed when a user initially submits a CR through GitHub/GitLab. The proposed CR is checked for adherence to coding standards, compiled, and tested. This preliminary testing is conducted on both the RAVEN software and RAVEN supported Plug-ins, and the test results are reported as comments and status updates on the Pull Request (within GitHub/GitLab). The tests are performed on various Operative Systems and configurations. - If there are syntax/compile errors or failures to comply with coding/commentary standards, the change will be rejected.
Code Review	RAVEN core Team	- All code changes go through a peer-review process prior to being merged, both to ensure correctness and to determine the appropriateness of the implemented design using the method described in PLN-5552. - To maintain independence, code proposed by any given developer must be reviewed and merged by someone other than the original CR author. After the Pull Request has passed the tests and one or more members of the RAVEN core team (*CCB* members) have verified the design, the Pull Request is merged into the development branch of the repository. - Code review is required for any modification (source code, documentation, etc.).
Development Branch Testing (RAVEN software)	Automated	- This step includes a suite of tests similar to but more extensive than the "Pull Request" testing.

		- The development branch of RAVEN software is tested against a specific version of the RAVEN supported Plug-ins. Failures of the Plug-ins tests will be noted and the developers of the Plug-ins will be informed about the failure for future reference and to take actions in addressing the problem (e.g. change in the API, etc.). - After the tests pass, the development branch is automatically merged into the master branch.
Development Branch Testing (RAVEN Plug-ins)	Automated	- The development branch of the RAVEN supported Plug-in is tested against the master branch of RAVEN software. - Test failures will cause the automated system to report a failure (and therefore prevent the Plug-in's development branch from merging into the master branch). - After the RAVEN supported Plug-ins tests pass, the development branch is automatically merged into the master branch.
Master Branch Testing	Automated	- Similar to the Development branch testing, but on the master (stable) branch to assure that all integration occurs without issue.
Pre-release Branch Testing	Technical Leader	- Testing that occurs in preparation of a new release (from the master branch) of the RAVEN software. In this branch, the full suite of tests (verification and validation) is performed automatically. - The Technical Leader is responsible to testify the correctness of the new features/capabilities introduced since the last release of the software. - If all the tests pass and the new features/capabilities get approved by the Technical Leader, the branch will be merged in release and a new release of the software is issued.

		- The Technical Leader is responsible to approve the merge into the release branch and to record in GitHub the approved new release.
Documentation Testing	Automated	- After the master branch is updated, the various documentation- related tasks are executed. These include updates to the *Doxygen* (see def.)-based source code documentation, XSD schema, and test coverage.

11.1 V&V Reporting Requirements

V&V reporting requirements are generally prescribed for any software development effort that qualifies as a project as defined in LWP-13620, "Managing Information Technology Assets." V&V reporting for M&O, while iteratively following the same development process as a project, is satisfied with the test reports associated with any documented change request (CR).

The following reporting requirements are to be addressed within the Project Management Plan, when its use is required:

- Requirement, design, and implementation review reports
- *Anomaly* (see def.) reports
- *System test* (see def.) report
- V&V final report or acceptance test report.

Considering the degree of automation that has been developed for the RAVEN software distribution, the above reporting requirements are handled in electronic format directly during the deployment of the code (GitHub).

Reporting the results from the execution of the automated tests is available from:

1) Verification and Validation Results Report:

The report about the verification and validation results is an online report generated during RAVEN software and RAVEN supported Plug-ins testing. The following information is automatically recorded in each test record: program tested (RAVEN or specific Plug-in), hardware tested, date, time, tester (automated), success or failure (details of failed tests), applicability (encoded automatically as test "step" result), actions taken for any deviations and by whom (as traceable through issue).

At the most detailed level, the results of the automated tests which are run at the various test process steps described in Table , are available for an individual "pull request." The overall pass/fail status of the test run at each step is available by pull request number through the GitHub/GitLab interface, and then the detailed results of each test are available through the CIS website. The results of each individual test step are displayed in CIS website, indicating whether the test passed or failed. This information may be used as a cursory review of the changes prior to performing a full review.

2) Test Coverage Report:

As part of the V&V process, the RAVEN core team monitors the percentage of code covered by the tests. The code coverage report is available on a protected server within Idaho National Laboratory (INL) (https://hpcsc.inl.gov/ssl/RAVEN/python-coverage/). The coverage report is reviewed on a periodic basis by the RAVEN core team to determine if new tests need to be created. If the coverage falls below 80%, the Technical leader is responsible to define the appropriate corrective actions that must be taken.

11.2 V&V Administrative Requirements

11.2.1 Anomaly Resolution and Reporting

All *software anomalies* (see def.) discovered either during the V&V effort or during the course of the deployment effort are captured in the *issue* (see def.) tracking system within GitHub/GitLab. When errors are found during testing, the issue entered into GitHub/GitLab is used to track the correction of the error, along with reporting its initial identification. Each issue captures the following information:

- Uniquely identifying issue number
- Detailed issue description.
- The date the error was discovered.
- The name of the person who discovered the anomaly.
- The software system with which the issue is associated (RAVEN software, RAVEN regression test system, Code Interfaces, Documentation, etc.)
- The issue type (tagged as a "*defect*" for anomalies, "task" for a suggested improvement).
- The issue priority (critical, normal, minor).

- Optional comment thread is available for the RAVEN core team to provide feedback.

- The problem resolution will be contained within a pull request associated with the issue. A pull request does not necessarily have to be associated with an active issue number. Details of the change are included in the pull request.

Once an issue is created (either for "*defect*" and "*task*"), a member of the CCB shall review it, following the "Issue Review" check-list (https://github.com/idaholab/raven/wiki/development-checklists#issue-review-checklist). A similar check-list (named "Issue Close" check-list) is followed by the reviewer of the pull request associated with the issue, upon its closure (https://github.com/idaholab/raven/wiki/development-checklists#issue-close-checklist).

Identified *anomalies* and proposed resolutions are documented and processed per the Problem Reporting and Corrective Action section 12.

11.3 Task Iteration Policy

The tasks as outlined in Table provide sufficient rigor to meet the INL QA program requirements. If the end-use or scope of the asset changes significantly, the criticality/risk analysis must be reviewed. If the results of the analysis change, the IT project manager or M&O manager reviews the adequacy of the V&V performance to determine if additional V&V activities and/or frequency need to be modified.

11.4 Control Procedures

The CIS is designed to generate and store the following metrics, used by the regression test system, to determine if the code is promoted to production or remediated:

- Pass/fail (primary criteria)

- Code coverage (must be 80% or higher)

Test evaluation, performed by an independent reviewer (member of the *CCB*), is achieved by viewing the results on the CIS either through GitHub/Gitlab or by browsing, in the CIS website, for specific CRs to ensure that the test requirements have been satisfied to demonstrate the capability of the software to provide valid results for test problems encompassing the range of documented permitted usage, as applicable. All RAVEN and RAVEN supported Plug-ins tests have acceptability criteria designed into the test by the test creator so that every test has a binary status (pass/fail). This effectively removes the burden from developers

and users running tests from having to determine if tests meet acceptability requirements. An assigned member of the *CCB* also evaluates the results. This evaluation is recorded by the repository management software (GitHub/GitLab) as actions are taken.

12. PROBLEM REPORTING AND CORRECTIVE ACTION

Problems and corrective actions within the scope of the RAVEN software and RAVEN supported Plug-ins are tracked using the CR process to report and resolve issues. For issues discovered during the verification process, resolution will be tracked on the original CR related to the software modification.

Any system user, internal or external, can report an issue in GitHub (for RAVEN software) or GitLab (for RAVEN supported Plug-ins). The issue is evaluated and categorized in the *Version Control System* (i.e. GitHub) as either a "Defect" - bug or "Improvement" - a non-critical task such as a feature enhancement; or the issue is closed without category if the issue was created accidentally or due to an imperfect knowledge of the software by the initiator of the Issue. Any internal or external user can choose to work on a particular issue simply by associating the issue with a pull request in GitHub. A listing of all active issues is available through GitHub for all users to review. A similar system exists internally for the RAVEN supported plugin-ins through GitLab.

If a problem relates to RAVEN and violates a requirement, the issue will be documented and managed in the INL Issues Management System, per LWP-13840, "Issues Management."

Any Issue categorized as "Defect" will require a notification to the users (email or group notification through the GitHub/GitLab software) upon resolution. A notification to the users can be avoided in case the Chair of the CCB or his designee approves the exception for justified reasons.

The process related to address problems/defects and consequential corrective actions are fully documented in PLN-5553.

13. TOOLS, TECHNIQUES, AND METHODOLOGIES

The RAVEN software and its supported Plug-ins are in continuous evolution, via an *Agile development process* (see def.), since new expansions and capabilities are needed by the different projects/programs. In order to guarantee the SQA standards identified by this plan, an articulated set of tools, techniques and methodologies are used as applicable:

Methods:

- Test and user-need driven development

- Pull requests
- Continuous integration.

Techniques:

- Code coverage analysis
- Regression testing
- Expected error testing
- Unit testing (when applicable)
- Cascading builds
- Agile development
- Peer reviews
- Performance testing
- Shared repository (GitHub)
- Decentralization.

Tools:

- GitHub (Git) software code repository used outside the INL software network.
- Python software development language
- C++/C language
- Bash scripting
- Wiki – RAVEN documentation
- Doxygen – Software framework documentation generator
- Enterprise Architecture (EA)
- EDMS Safety Software Determination system
- Quality Level Determination (QLD) system.

14. SUPPLIER 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.

15. RECORDS COLLECTION, MAINTENANCE, AND RETENTION

The primary quality assurance (QA) records include:

- Management plans
- Business and technical requirements
- Design documentation
- Independent review documentation
- Verification and validation results
- Change control and configuration management documentation
- Assessment reports.

All RAVEN and RAVEN supported Plug-ins QA records are managed, and retention periods set per PLN-4653, "INL Records Management Plan." Retention periods for electronic records (e.g. GitHub) are identified and tracked in the Asset Portfolio per LWP-1202.

16. TRAINING

Project manager is responsible for ensuring implementation of the required SQA and training.

The RAVEN core team personnel have been selected based on the expertise required for RAVEN software development.

Training includes the following activities, all of which will be documented on the employee's individual training plan within the TRAIN system:

- Complete laboratory IT Asset Management training course 0INL1631
- Orientation for this plan will be given to every staff member by the RAVEN core team's project manager or designee.

- Team members will also be trained in the use of Git, GitHub, GitLab, coding, and commentary standards through study of the RAVEN Developer Guide webpage (https://github.com/idaholab/raven/wiki/Developer_Information).

Required training shall be implemented as described in PDD-13610, "Software Quality Assurance Program."

17. RISK MANAGEMENT

The risk analysis for each application is documented on the safety software determination (SSD) and quality level determination (QLD). The SSD and QLD are identified in the EA repository for each individual application. Risks associated with the RAVEN software and RAVEN supported Plug-ins are controlled via the rigor implemented in requirements identification, testing, verification and validation, and change control processes.

17.1 Safety Software Determination

The SSD documents the decision basis as to why a software application is or is not safety software. The record copy is maintained within the company approved electronic document management system in accordance with LWP-13014, "Determining Quality Levels."

The RAVEN software and RAVEN supported Plug-ins will be required to have a documented SSD. The SSD for RAVEN supported Plug-ins will be re-evaluated in case of moving to operation.

17.2 Quality Level Determination

The QLD documents the risk analysis in accordance with LWP-13014, based on the end use of the RAVEN software and supported Plug-ins. The QLD for RAVEN supported Plug-ins will be re-evaluated in case of moving to operation.

18. ASSET MAINTENANCE

The M&O activities shall be performed following the same development process used for adding new capabilities (i.e. Agile development process). M&O activities will be performed by the RAVEN core or RAVEN supported Plug-ins teams to ensure continuity and homogeneity of the software.

18.1 Business Requirements

The business requirements associated with the RAVEN software and RAVEN supported Plug-ins coincide with the software requirements specified in SPC-2366, "RAVEN Software Requirements Specification." The software requirements are stored and maintained in the RAVEN *GitHub* (see def.), for

open-source software, and *GitLab* (see def.), for supported (and not open-source) Plug-ins.

18.2 Schedule and Budget and Summary

The budget for the RAVEN software and RAVEN supported Plug-ins (for M&O and development) is fluid and changing based on year to year customer needs and funding granted by research and development activities. The current year budget can be obtained by contacting the responsible project manager or asset owner.

18.3 Evolution of the Plan

The M&O manager is responsible for maintaining this plan and ensuring that the M&O activities necessary are appropriately executed throughout the life cycle of the RAVEN and RAVEN supported Plug-ins.

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 and approved by the RAVEN Asset Owner.

18.4 System Hardware and Operating Systems

There are no restrictions regarding the type of computing hardware and operating systems that may be used, provided it can run the development tools for the application software. For internal INL personnel, anomalies identified with hardware will be reported to the Ops Center (6-1000), and the INL Field Services organization will be utilized to ensure the identified hardware anomalies are resolved.

18.5 Backup and Recovery

RAVEN software and RAVEN supported Plug-ins are stored on servers within and outside of the INL network. The servers containing RAVEN software code and its supported Plug-ins are identified in the EA. The main repository for RAVEN source code is hosted by GitHub and can be found at https://github.com/idaholab. The GitHub service also functions as a system for generating tracking *issues* (see def.) that are used as the main form of a *change request* (see def.). The tracking issues and associated data contained within the external GitHub services are backed up to local services at least once a year. The GitHub service is hosted in a cloud environment and features its own backup and recovery protocols offered by the managing services.

Local copies of the RAVEN software code are kept on the computer of each RAVEN core team member. In addition, business continuity is managed in

accordance with PLN-117, "Information Management Contingency Plan for the Idaho National Laboratory."

19. MAINTENANCE AND OPERATIONS PLANNING

19.1 M&O Initiation

RAVEN software and RAVEN supported Plug-ins, covered by this plan, is presently in the M&O phase. This section specifies the details for initiating the transition to M&O including estimation of the required staffing, training, and other resources to support the M&O activity.

The M&O manager is responsible for ensuring adherence to this plan.

19.1.1 Estimation Plan

The budget for conducting M&O is fluid and changing based on year to year customer needs and funding granted by research and development activities. 0.2/0.3 FTEs are generally appropriate for M&O activities.

19.1.2 Staffing Plan

The number of RAVEN core team members is fluid and changing based on year to year needs and customer expectations. The staffing is performed by the RAVEN project manager in accordance with the Technical Leader.

19.1.3 Training Plan

Project manager is responsible for ensuring implementation of the required SQA and training for both deployment and M&O activities.

The RAVEN core team personnel have been selected based on the expertise required for RAVEN software development.

Personnel assigned to any of the roles supporting the RAVEN software shall be assigned Training Records and Information Network (TRAIN) job code IRSBNL0000, IT Asset Management, on their employee training plan.

Training includes the following activities, all of which will be documented on the employee's individual training plan within the TRAIN system:

- Orientation for this plan will be given to every staff member by the RAVEN core team's project manager or designee.

- Team members will also be trained in the use of Git, GitHub, GitLab, coding, and commentary standards through study of the RAVEN Developer Guide webpage (https://github.com/idaholab/raven/wiki/Developer_Information).

Required training shall be implemented as described in PDD-13610, "Software Quality Assurance Program."

20. M&O Work Plans

20.1 Work Activities

The following M&O work activities are performed by the RAVEN core team:

- System administration
- Change management
- Requirements analysis
- Design
- Development
- Verification and validation
- Configuration management

Configuration management and change control activities must be performed in accordance to Section 4 of this plan. Verification and validation and requirements analysis activities are governed by Section 14 of this plan. Software quality assurance is guided by this plan.

20.2 Resource Allocation

Under the guidance of the RAVEN Asset manager, resource allocation is made by the M&O manager and/or RAVEN Technical lead. Most of the assigned resources for RAVEN M&O work activities are for development. Each assigned developer participates in each of the other identified work activities identified in Table 1 of this plan, due to the nature of the *Agile development* process, decentralization and continuous integration methods taken to maintain the RAVEN software.

The resource allocation is performed at needs-basis.

20.3 Budget Allocation

The RAVEN software maintenance is supported by the different programs/projects that use the software. The budget allocation is performed on yearly basis. For the current budget allocation, refer to the Asset owner.

For aspects of the RAVEN software that require budget allocations, refer to the EA repository.

20.4 Acquisition Plan

The asset owner will acquire materials and services that are necessary to support M&O activities for RAVEN 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).

Acquisitions must be handled in accordance with the following procedures:

- LWP-4001, "Material Acquisitions"
- LWP-4002, "Service Acquisitions"
- LWP-1305, "Acquisition of Computer Hardware/Software Resources."

When acquiring application software (including upgrades), the following documentation is required:

1. Business requirements describing the capabilities and limitations.
2. Test plans and test cases that will be used to validate the capability of the system for its specific application.
3. Instructions for use.

21. M&O ASSESSMENT AND CONTROL

This section specifies how the M&O team will assess and control the product requirements/design as well as the quality and timeliness of acquired products from subcontractors.

21.1 Requirements and Design Control Plan

Changes in scope are documented with associated impacts to the requirements and design, schedule, and budget. Requirement and design documentation shall be re-baselined and approved on a biannual basis.

21.2 Subcontractor Management Plan

At this time, there are no plans to utilize subcontractors for RAVEN software and its supported Plug-ins M&O activities. If a need to assign subcontractor resources arises in the future, this document will be updated to include Subcontractor/Vendor subcontract and vendor management

22. SUPPORTING PROCESS PLANS

This section contains plans for the supporting processes that span the duration of the M&O activity.

22.1 Communication and Publicity

Due to the open source nature of the RAVEN software framework, communication is part of the process that takes place during M&O work activities.

The documentation generation activities, Wiki, and GitHub/GitLab available to RAVEN software framework users are sufficient methods to communicate successfully integrated RAVEN software changes.

In addition, all the stakeholders listed in this plan shall receive information about M&O activities, in case of raised concerns or changes of this plan and subordinate plans.

The information, status and deviations will be shared via the RAVEN user and development email lists.

22.2 Assessments

All elements of the INL QAP are described in PDD 13000 and evaluated on a 3-year cycle. The SQA element, including the processes and training used for software development, performance, and maintenance, is included for evaluation in this 3-year cycle.

22.3 Retirement

When the RAVEN core team determines that the RAVEN software (or one or more supported Plug-ins), addressed by this plan has reached end-of-life, the retirement plan must be documented and approved by the asset owner. In addition, the following activities must be completed, as applicable:

- Electronic data, if any, will be transferred to a replacement system or archived per the records disposition as specified on the records analysis stored in the Enterprise Architecture Repository.

- This plan and any associated controlled documents in EDMS will be updated to reflect the change in asset disposition. If all assets within the scope of this plan are retired, this plan and all associated controlled documents in EDMS will be cancelled and records destroyed.

- Access to the identified asset will be terminated.

- The identified asset will be removed from both INL and external network infrastructure.

The status for the EA repository record for identified asset will be changed to "retired."