|  |
| --- |
| Document ID: PLN-5554  Revision ID: 0  Effective Date: 01/10/2018 |
| Software Test and V&V Plan |
|  |
| RAVEN and RAVEN Plug-ins |
|  |
|  |
| RAVEN Software and RAVEN Plug-ins Test and V&V Plan |

|  |  |  |
| --- | --- | --- |
| RAVEN Software and RAVEN Plug-ins Test and V&V Plan | | |
| PLN-5554 | | |
|  | | |
| **Prepared by:** |  | |
| [Required. For EDMS controlled documents, reference the eCR.] |  |  |
| IT Project/M&O Manager |  | Date |
| **Reviewed by:** |  | |
| [Required for QL-1 and QL-2. For EDMS controlled documents, reference the eCR.] |  |  |
| Independent Reviewer |  | Date |
| **Approved by:** |  | |
| [Required. For EDMS controlled documents, reference the eCR.] |  |  |
| IT Asset Owner |  | Date |

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

CONTENTS

[1. TEST AND V&V SCOPE 4](#_Toc525904191)

[1.1 Specific meaning of V&V activities for RAVEN software 5](#_Toc525904192)

[2. REFERENCES 5](#_Toc525904193)

[3. DEFINITIONS AND ACRONYMS 5](#_Toc525904194)

[3.1 Definitions 5](#_Toc525904195)

[3.2 Acronyms 9](#_Toc525904196)

[4. ASSUMPTIONS 10](#_Toc525904197)

[5. CONSTRAINTS 10](#_Toc525904198)

[6. TEST AND V&V OBJECTIVES 10](#_Toc525904199)

[7. TYPES OF TESTS TO BE EXECUTED 11](#_Toc525904200)

[8. APPROVAL REQUIREMENTS 13](#_Toc525904201)

[8.1 Requirement tests 13](#_Toc525904202)

[8.1 Other tests 13](#_Toc525904203)

[9. TEST AUTOMATION 14](#_Toc525904204)

[10. RESOURCE REQUIREMENT 14](#_Toc525904205)

[10.1 Human Resources 14](#_Toc525904206)

[10.2 Hardware/Software Resources 14](#_Toc525904207)

[10.3 Services/Applications 15](#_Toc525904208)

[11. TEST DEFINITION TASKS AND RESPONSIBILITIES 15](#_Toc525904209)

[12. V&V PROCESSES 15](#_Toc525904210)

[13. V&V Reporting Requirements 18](#_Toc525904211)

[14. V&V Administrative Requirements 19](#_Toc525904212)

[14.1 Anomaly Resolution and Reporting 19](#_Toc525904213)

[14.2 Task Iteration Policy 20](#_Toc525904214)

[14.3 Deviation Policy 20](#_Toc525904215)

[14.4 Control Procedures 20](#_Toc525904216)

[15. Plan Maintenance 21](#_Toc525904217)

# 

# TEST AND V&V SCOPE

As stated in “RAVEN Software Quality Assurance Plan”, PLN-5552, RAVEN is open source software that is maintained and stored in a public repository (GitHub, see def.). 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* (see def.) both in *GitHub*, for open source software, and in GitLab, 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 (i.e., unit, integration, system, deployment) are covered by the RAVEN framework and, optionally, each RAVEN Plug-in.

The project manager/technical leader oversees the testing and 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, and it is determined at that step in the process of 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 (e.g., automated test scripts, fault insertion tools, etc.);
* Type and extent of required testing; and
* Required reviews and approvals.

A component of the *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.

The *Continuous Integration System* will verify that the provided documentation ensures that the software demonstrates adherence to the documented requirements and that the software produces correct results.

## Specific meaning of V&V activities for RAVEN software

As stated in “RAVEN Software Quality Assurance Plan”, PLN-5552, RAVEN is a multi-purpose UQ, 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* of the RAVEN software is mostly related to the verification of the models/capabilities with analytical testing or process (when applicable).

# REFERENCES

ISO/IEC/IEEE 24765:2010(E), “Systems and software engineering — Vocabulary,” First Edition, December 15, 2010.

LWP‑1201, “Document Management,” Rev. 10, May 1, 2013.

LWP‑13620, “Managing Information Technology Assets,” Rev. 16, December 23, 2013.

# DEFINITIONS AND ACRONYMS

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

## Definitions

*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.

*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]

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

*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.

*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, Jenkis, and GitLab Continuous Integration.

*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 the applications/extensions/plugins 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.

*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]

*RAVEN core team.* INL personnel whose job description includes the development of the RAVEN software or software applications/extensions/plugins that are based on the RAVEN framework.

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

*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.

*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 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.*

* 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.
* Formal proof of program correctness (e.g., requirements, design, implementation reviews, system tests). [ISO/IEC/IEEE 24765:2010(E) edited]

*Custom‑developed software.* Software built specifically for an INL application. It may be developed by INL 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. [DOE G 414.1‑4 edited]

*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 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)]

*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.

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

## Acronyms

ASME American Society of Mechanical Engineers

CCB Change Control Board

CM Configuration Management

CR Change Request

CSV Comma Separated Value

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

POSIX Portable Operating System Interface

PRA Probabilistic Risk Assessment

QA Quality Assurance

QLD Quality Level Determination

RAVEN Risk Analysis and Virtual ENviroment

SQA Software Quality Assurance

SSD Safety Software Determination

UQ Uncertainty Quantification

V&V Verification and Validation

# ASSUMPTIONS

The testing of the RAVEN software and its supported Plug-ins is performed automatically (see Section 9) for each *CR* (see def.) by the *Continuous Integration System* (see def.). It is assumed that the verification and testing suite (test cases) is able to test the main attributes and any combination of any new added capability.

# CONSTRAINTS

The RAVEN software and RAVEN supported plug-ins are constrained to be executed on any POSIX compliant system (including Windows POSIX emulators such as MinGW).

No other constraints are envisioned.

# TEST AND V&V OBJECTIVES

The software test and V&V plan for the RAVEN software and RAVEN Plug-ins is developed in accordance with LWP-13620. The plan shall include 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 “RAVEN Configuration Management Plan, PLN-5553”. The responsibilities and tasks for modifying/adding test cases associated with one or multiple requirements is documented in Section 8.

# 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 Continuous Integration System tests that the User Documentation and SQA Documentation can correctly be generated.
* XSD Schema Validation: The Continuous Integration System tests that all the test files are compliant with the prescribed input syntax.
* Code Standard Validation: The Continuous Integration System 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 Continuous Integration System tests that at least the 80% of the source code is tested by the test suite.

# APPROVAL REQUIREMENTS

The RAVEN and RAVEN 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:

## Requirement tests

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

Table 1 - Requirement tests' responsibilities.

|  |  |
| --- | --- |
| **Test Case Reviewer(s):** | Chair of the *CCB*, Technical Leader, Independent Reviewer (Member of the *CCB*) |
| **Test Result Reviewer and Approver:** | Chair of the *CCB*, Technical Leader, Independent Reviewer (Member of the *CCB*) |
| **Acceptance Test Case Reviewer(s):** | Chair of the *CCB*, Technical Leader, Independent Reviewer (Member of the *CCB*) |
| **Acceptance Result Reviewer(s):** | Automated Continuous Integration System |
| **Acceptance Result Approver:** | Automated Continuous Integration System |

## 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 2 - 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 Continuous Integration System |
| **Acceptance Result Approver:** | Automated Continuous Integration System |

# TEST AUTOMATION

Testing is performed automatically as part of the Continuous Integration System 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 will be 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* (see def.) and *GitLab* (see def.) for protected 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 *Continuous Integration System* database, in an approved records repository along with results from the timing executions and code coverage.

# RESOURCE REQUIREMENT

As mentioned, the testing and acceptance requirement approval is performed in an automated fashion in multiple operative systems and for each CR.

## Human Resources

No human resources are required for day by day testing, since the automated fashion of the testing system.

As mentioned in Section 6, any developer is required to provide a new test or modify an existing one for covering the functionality of new or modified capabilities (CR).

## Hardware/Software Resources

RAVEN and RAVEN Plug-ins are automatically tested in dedicated machines. Resources have been allocated at the begin of the project for setting up those machines.

## Services/Applications

The RAVEN software and RAVEN Plug-ins make use of *Continuous Integration Systems* (see def.) for automating the testing and approval of acceptance criteria. These tools are already integrated in the version control repositories GitHub and GitLab.

# TEST DEFINITION TASKS AND RESPONSIBILITIES

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

Table 3 - Tasks and responsibilities for tests creation.

| **Tasks** | **Responsibility** |
| --- | --- |
| 1. Complete programming and test creation | Developer of the proposed CR |
| 1. Test data creation | Developer of the proposed CR |
| 1. Set up test environment | Automated via *Continuous Integration System* |
| 1. Migrate services to test environment | Automated via *Continuous Integration System* |
| 1. Set up test database | Automated via *Continuous Integration System* |
| 1. Prepare test cases | Developer of the *CR* |
| 1. Conduct test, record results, and communicate to the developers | Automated via *Continuous Integration System* |
| 1. Make corrections and updates to the processes | Developer of the *CR* |
| 1. 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*

# V&V PROCESSES

The V&V tasks, both automatic and manual, occur for the RAVEN software and RAVEN 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. These tasks are listed in the Table below.

Table 4 – 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 (see def.). 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 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 “RAVEN and RAVEN Plug-ins Software Quality Assurance Plan”, Software Reviews, 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 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 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. |

# 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.

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). These test records are readily available online and to the *CCB* during the review process discussed in Section 14.4.

At the most detailed level, the results of the automated tests which are run at the various test process steps described in Table 4, 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 *Continuous Integration System* website. The results of each individual test step are displayed in *Continuous Integration System* 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.

1. **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 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.

# V&V Administrative Requirements

## 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, “improvement” 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 the 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 (both 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 of “RAVEN Software Quality Assurance Plan”, PLN-5552.

## Task Iteration Policy

The tasks as outlined in Table 4 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.

## Deviation Policy

Planned deviations from this plan require management approval. Unplanned deviations, upon discovery, will require identification, rationale, and the estimated effect on software quality.

Deviations that violate requirements must be documented within the laboratory issues and corrective action management system and reviewed and approved by QA and the same individuals who approved this plan.

## Control Procedures

The *Continuous Integration System* 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 *Continuous Integration System* either through *GitHub/Gitlab* or by browsing, in the *Continuous Integration System* 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 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.

# Plan Maintenance

The RAVEN project manager or M&O manager is responsible for maintaining this plan and ensuring that the activities necessary for V&V are appropriately executed throughout the life cycle of the RAVEN software and supported RAVEN 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 or on a biannual basis. Modifications to this plan must be independently reviewed and approved by the RAVEN Owner.