|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Laboratory-wide | | Template | |  |  |  | |  | |
|  |  | | | | | | | | |
| Document ID: PLN-6552  Revision ID: 0  Effective Date: 02/24/2022 | | | | | |
| Software Quality Assurance Plan (SQAP) | | | | | |
|  | | | |  | |
|  | | | |  | |
|  | | | | | |
|  | | | |  | |
| FORCE Software Quality Assurance and Maintenance and Operations Plan | | | | | | | |

|  |  |  |
| --- | --- | --- |
| FORCE Software Quality Assurance and Maintenance and Operations Plan | | |
| PLN-6552 | | |
|  | | |
| **Prepared by:** |  | |
| Konor Frick |  | 02/24/2022 |
| IT Project/M&O Manager |  | Date |
| **Reviewed by:** |  | |
| Konor Frick |  | 02/24/2022 |
| Independent Reviewer |  | Date |
| **Approved by:** |  | |
| Konor Frick |  | 02/24/2022 |
| Asset Owner |  | Date |

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

CONTENTS

[1. PURPOSE 3](#_Toc97539985)

[1.1 FORCE Description 3](#_Toc97539986)

[1.2 Software Lifecycle 4](#_Toc97539987)

[1.3 Assumption and Constraints 4](#_Toc97539988)

[1.4 Deviation Policy 4](#_Toc97539989)

[2. REFERENCES 5](#_Toc97539990)

[3. DEFINITIONS AND ACRONYMS 6](#_Toc97539991)

[3.1 Definitions 6](#_Toc97539992)

[3.2 Acronyms 12](#_Toc97539993)

[4. MANAGEMENT 14](#_Toc97539994)

[5. CONFIGURATION MANAGEMENT 14](#_Toc97539995)

[6. SUBCONTRACTOR.VENDOR 14](#_Toc97539996)

[7. DOCUMENTATION 14](#_Toc97539997)

[7.1 Minimum Documentation Requirements 14](#_Toc97539998)

[7.2 Other Documentation 15](#_Toc97539999)

[8. STANDARDS, PRACTICES, CONVENTIONS, AND METRICS 15](#_Toc97540000)

[8.1 Content 15](#_Toc97540001)

[8.1.1 Software Coding Standards 15](#_Toc97540002)

[8.1.2 Commentary Standards 16](#_Toc97540003)

[8.1.3 Testing Standards and Practices 16](#_Toc97540004)

[9. SOFTWARE REVIEWS 16](#_Toc97540005)

[10. TESTING 16](#_Toc97540006)

[10.1 V&V Overview 16](#_Toc97540007)

[10.1.1 Test & V&V Objectives 16](#_Toc97540008)

[10.1.2 Master Schedule 18](#_Toc97540009)

[10.1.3 Specific meaning of V&V activities for FORCE software 19](#_Toc97540010)

[10.2 TYPES OF TESTS TO BE EXECUTED 19](#_Toc97540011)

[10.3 Test Automation 20](#_Toc97540012)

[10.4 APPROVAL REQUIREMENTS 21](#_Toc97540013)

[10.5 Requirement tests 21](#_Toc97540014)

[10.6 Other tests 21](#_Toc97540015)

[10.7 TEST DEFINITION TASKS AND RESPONSIBILITIES 22](#_Toc97540016)

[11. V&V PROCESSES 22](#_Toc97540017)

[12. PROBLEM REPORTING AND CORRECTIVE ACTION 22](#_Toc97540018)

[13. TOOLS, TECHNIQUES, AND METHODOLOGIES 23](#_Toc97540019)

[14. SUPPLIER CONTROL 23](#_Toc97540020)

[15. RECORDS COLLECTION, MAINTENANCE, AND RETENTION 23](#_Toc97540021)

[16. TRAINING 23](#_Toc97540022)

[17. RISK MANAGEMENT 23](#_Toc97540023)

[17.1 Safety Software Determination 23](#_Toc97540024)

[17.2 Quality Level Determination 24](#_Toc97540025)

[18. ASSET MAINTENANCE AND MAINTENANCE AND OPERATIONS PLANNING 24](#_Toc97540026)

[19. M&O Work Plans 24](#_Toc97540027)

[20. M&O ASSESSMENT AND CONTROL 24](#_Toc97540028)

[21. SUPPORTING PROCESS PLANS 24](#_Toc97540029)

# 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 FORCE Software. 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 FORCE. 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”. This plan is based on the RAVEN SQA process, documented in “PLN-5552, *RAVEN and RAVEN Plug- ins Software Quality Assurance and Maintenance and Operations Plan*”. The FORCE software process follows the PLN-5552 and in this document, the deviations from such plan are documented.

## FORCE Description

The Framework for Optimization of ResourCes and Economics is a collection of software tools, models, and datasets acquired and developed under the Integrated Energy Systems (IES) program to enable analysis of technical and economic viability of myriad IES configurations. FORCE is the consolidating interface and data repository for all the IES toolsets ranging from macro technoeconomic analysis to transient process modeling and experimental validation for integrated energy systems.

FORCE will provide a central analyst-friendly experience to quickly identify top IES technology contenders for target regions and end use application. Once top contenders are identified analysis can be extended within the platform for intimate coupling, control, deployment, and experimental validation. If successful FORCE provides a holistic approach to IES analysis at all levels, scoping to deployment.

FORCE is operational within multiple projects. Ongoing support of FORCE is required for the purpose of adding functionality, improving the user experience, updating use cases and economic costing functions.

* FORCE is maintained by a team of scientists/researchers, referred to herein as the FORCE core team (see def.). FORCE maintenance, performed by the FORCE core team, is an ongoing activity.
* This plan covers the maintenance of all existing and future components of FORCE. This includes, but is not limited to, servers, server software, user workstations, FORCE software, and control documents. Changes to this document will be completed through the Electronic Change Request (eCR) process.

## Software Lifecycle

FORCE 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.

## Assumption and Constraints

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

## 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.

# 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-5552, “RAVEN and RAVEN Plug- ins Software Quality Assurance and Maintenance and Operations Plan”
* PLN-4653, “INL Records Management Plan”
* SDD-584, “FORCE Software Design Description (SDD)”
* SPC-3171, “FORCE Software Requirements Specification (SRS) and Traceability Matrix”

# DEFINITIONS AND ACRONYMS

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

## Definitions

*Acquired software.* Software generally supplied through basic procurements, two- party agreements, or other contractual arrangements. Acquired software includes commercial off-the-shelf software, support software such as operating systems, database management systems, compilers, software development tools, and commercial calculational software and spreadsheet tools (e.g. 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.

*Dymola*. Dymola is a commercial modeling and simulation environment based on the open Modelica modeling language, Developed by the European company Dassault Systèmes.

*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.

*FMI.* The Functional Mock-up Interface (or FMI) defines a standardized interface to be used in computer simulations to develop complex cyber-physical systems.

*FMU.* Based on an FMI, the FMU is an executable called a Functional Mock-up Unit (FMU), which is “driven” by an FMI. A simulation environment can use the FMI to create an instance of the FMU and simulate it together with other FMUs or models native to the simulation environment.

*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. 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.

*Modelica.* Object-oriented, declarative, multi-domain modeling language for component-oriented modeling of complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

*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>

*FORCE core team.* INL personnel who are in charge of the development of the FORCE software applications/extensions that are based on the FORCE software. A list of the core team will be made available on Github.

*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.

*FORCE Software. Collection of software/models/analysis workflows* that resides in a public repository (soon to be GitHub) that provides the top level interface for performing technoeconomic and transient analysis of Integrated Energy Systems from INL with example workflows demonstrating the capabilities.

*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].

*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.)

## 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

FMI Functional Mock-up Interface

FMU Functional Mock-up Unit

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

# MANAGEMENT

The MANAGEMENT plan of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# CONFIGURATION MANAGEMENT

The CONFIGURATION MANAGEMENT plan of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE). The FORCE configuration items’ list can be found in LST-1296.

# SUBCONTRACTOR.VENDOR

No subcontractors/vendors activities are envisioned for FORCE Software. In case of a new strategy, involving subcontractors, is defined, this plan will be revised.

# 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 FORCE, the following documents will be maintained as part of the Asset Portfolio.

## Minimum Documentation Requirements

As a minimum, the following documentation is required for the FORCE software. These documents are managed as records in accordance with Section 20, “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-6552 |
| Software Test Plan and Verification & Validation | GitHub | PLN-6552 |
| Software Requirements Specification and Traceability Matrix | GitHub | SPC- 3171 |
| Software Design Description | GitHub | SDD-584 |

## 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-000786, “FORCE Safety Software Determination”
* QLD, ALL-000916 “FORCE Quality Level Determination”
* FORCE Enterprise Architecture [Entry](https://ctm.inl.gov/ng/ctm/tabs/B3D94358-2BC6-46E8-8A7A-9367770A748C/E3232AB1-CF1E-4EF8-BC56-EFE1C1BE6526/search/details/6A036664-F7CA-4E6A-9CED-163A6237388B/cb6b): APM0001976

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.”

# STANDARDS, PRACTICES, CONVENTIONS, AND METRICS

## Content

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

The FORCE standards evolve around the following macro-areas:

* Software Coding Standards
* Commentary Standards
* Testing Standards and Practices

### Software Coding Standards

The FORCE software imposes a coding standard on all source code within the repository in line with the INL RAVEN and HERON repositories. 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.

### Commentary Standards

The FORCE software imposes a commentary standard on all source code within the repository in line with the INL RAVEN and HERON repositories. 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.

### Testing Standards and Practices

The FORCE software imposes a testing standard and practices on all the capabilities/methods of the FORCE software in line with the INL RAVEN and HERON repositories. This standard is publicly maintained on the FORCE 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.

# SOFTWARE REVIEWS

The SOFTWARE REVIEWS process of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# 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.

## V&V Overview

### 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.

FORCE 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.) in GitHub.

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 FORCE software.

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 *FORCE 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 *FORCE 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 HYBRID wiki page ([<https://github.com/idaholab/HYBRID>/-/wikis/Developing-Regression-Tests](https://hpcgitlab.inl.gov/hybrid/hybrid/-/wikis/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.

### Master Schedule

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

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

The FORCE software contains modelica models that will be, if available, compared with experimental results.

## 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 HYBRID wiki page ([<https://github.com/idaholab/HYBRID>/-/wikis/Developing-Regression-Tests](https://hpcgitlab.inl.gov/hybrid/hybrid/-/wikis/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.
* DymolaMatDiff: A test case that runs a simulation, terminates without error, and produces a previously defined “. mat” solution file within a predefined tolerance (usually to at least single precision accuracy or better).
* 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.

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.
* 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.

## 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 HYBRID software.

Use of the automated tests is integrated directly into GitHub, and as such does not require additional training other than general familiarity with performing a pull request in GitHub.

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.

## APPROVAL REQUIREMENTS

The FORCE software relies 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 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 |

## 

## 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 |

## 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 |
| 1. Test data creation | Developer of the proposed CR |
| 1. Set up test environment | Automated via CIS |
| 1. Migrate services to test environment | Automated via CIS |
| 1. Set up test database | Automated via CIS |
| 1. Prepare test cases | Developer of the *CR* |
| 1. Conduct test, record results, and communicate to the developers | Automated via CIS |
| 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 PROCESSES of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# PROBLEM REPORTING AND CORRECTIVE ACTION

The PROBLEM REPORTING AND CORRECTIVE ACTION of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# TOOLS, TECHNIQUES, AND METHODOLOGIES

The TOOLS, TECHNIQUES, AND METHODOLOGIES of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# SUPPLIER CONTROL

No subcontractors/vendors activities are envisioned for FORCE. In case of a new strategy, involving subcontractors, is defined, this plan will be revised.

# RECORDS COLLECTION, MAINTENANCE, AND RETENTION

The RECORD COLLECTION, MAINTENANCE, AND RETENTION process of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# TRAINING

The TRAINING process of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# 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 HYBRID software are controlled via the rigor implemented in requirements identification, testing, verification and validation, and change control processes.

## 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 FORCE software will be required to have a documented SSD.

## Quality Level Determination

The QLD documents the risk analysis in accordance with LWP‐13014, based on the end use of the HYBRID software.

# ASSET MAINTENANCE AND MAINTENANCE AND OPERATIONS PLANNING

The ASSET MAINTENANCE and MAINTENANCE AND OPERATIONS PLANNING processes of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# M&O Work Plans

The M&O Work Plans’ process of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# M&O ASSESSMENT AND CONTROL

The M&O ASSESSMENT AND CONTROL process of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).

# SUPPORTING PROCESS PLANS

The SUPPORTING PROCESS PLANS of the FORCE Software fully adheres with the one spelled out in the *PLN-5552, “RAVEN and RAVEN Plug-ins Software Quality Assurance and Maintenance and Operations Plan”* (replacing the word RAVEN with FORCE).