SANDIA REPORT

SAND2016-0765
Unlimited Release
Printed January 2016

SWiFT Software Quality Assurance Plan

Jonathan C. Berg

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: http://www.ntis.gov/search

SAND2016-0765
Unlimited Release
Printed January 2016

SWiFT Software Quality Assurance Plan

Jonathan C. Berg
Wind Energy Technologies Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS1124

Abstract

This document describes the software development practice areas and processes which contribute to the ability of SWiFT software developers to provide quality software. These processes are designed to satisfy the requirements set forth by the Sandia Software Quality Assurance Program (SSQAP).

APPROVALS

SWiFT Software Quality Assurance Plan (SAND2016-0765) approved by:

Department Manager		SWiFT Site Lead	
Dave Minster (6121)	Date	Jonathan White (6121)	Date

SWiFT Controls Engineer			
Jonathan Berg (6121)	Date		

CHANGE HISTORY

Issue	Date	Originator(s)	Description
A	2016/01/27	Jon Berg (06121)	Initial release of the SWiFT Software Quality Assurance Plan

ACKNOWLEDGMENTS

This document makes use of the process examples provided by the Sandia National Laboratories Advanced Simulation and Computing (ASC) Software Quality Support Team.

The initial SWiFT turbine controller was written by Vestas Technology R&D in collaboration with Sandia National Laboratories Wind Energy Technologies department.

CONTENTS

FIGURES

DEFINITION OF TERMS

Archive: 1) (verb) to store software configuration items, issues, and/or releases in a repository or shared location; 2) (noun) the repository or shared location containing software configuration items, issues, and/or releases.

Artifact: one of many types of work products resulting from the software development process. Design artifacts include documentation and design review records. Implementation artifacts include work products such as software code, user documentation, developer's guide, theory manual, and interface manual. Project planning artifacts include project plan and budget.

Baseline: 1) (noun) a set of software configuration items that serves as the basis for further development; 2) (verb) to capture a snapshot of a set of software configuration items at a reference point within the promotion model lifecycle.

Configuration Management: a discipline of identifying software configuration items and applying version control, tracking issues, labeling releases, and providing backup and recovery of data. Establishes and maintains a controlled repository (a baseline) of all artifacts created or collected during the software engineering process.

Release: (1) (noun) the contents of a baseline including executables and build instructions, if applicable. A release may be what is sent to an end-user, installed on a server, or provided for testing, and can be reproduced. A release can consist of all the files from the baseline from which it is created or it can consist of a subset of those baselined files. A release may also contain executables or build instructions; 2) (verb) to create the physical copy of the components contained within a baseline or built executables from the components of a baseline and make it available for installation or distribution.

Repository: a shared storage location where the archives for version-controlled files, issues, and/or releases are kept.

Revision: a recorded change to a software configuration item. Each time a user modifies a file and checks it back into a tool (cvs, Version Manager, etc.) the tool creates a new revision and assigns it a new revision number.

Version control: the process of recording revisions to software configuration items.

Version Control System: a central repository from which project deliverables can be managed. Typically, a version control system assigns revision numbers to each incremental change of a deliverable for tracking purposes. A version control system may also allow deliverable revisions to be tied to a general software release level, so that changes in software releases are traceable back to the deliverable revisions.

1. INTRODUCTION

This document describes the software development practice areas and processes which contribute to the ability of SWiFT software developers to provide quality software. These processes are designed to satisfy the requirements set forth by the Sandia Software Quality Assurance Program (SSQAP). A summary of SSQAP Required Practice Activities is given in Appendix A.

The software products in use at SWiFT include the following components, some of which are developed and maintained by Sandia and some of which are third-party software components which must also be considered in the software quality processes.

Sandia-maintained Software Components

- Turbine Controller with integrated data acquisition
- Met Tower data acquisition software
- SWiFT Control user interface for turbines and met towers
- Modifications to the Veristand Scan Engine and EtherCAT custom device for ABB ACS800 drive EtherCAT support
- FTP data file transfer program (from data source to control building)
- RMFT SWiFT-to-SNLNM database transfer routine
- ABB ACS800 drive parameter configuration
- Data analysis routines

Third-party Software Components

- National Instruments cRIO base software (Pharlap operating system & Veristand engine)
- National Instruments Veristand Add-on: Scan Engine and EtherCAT custom device[1]
- ABB ACS800 drive-side firmware and line-side firmware

[1] https://decibel.ni.com/content/docs/DOC-15510

2. SUMMARY OF SOFTWARE DEVELOPMENT, RELEASE, AND DEPLOYMENT

Software Development Process

1. Analyze requirements, create high level design, and assign tasks
2. Identify software implementation approach (what algorithms, how to meet software specification) and create detailed design
3. Document the detailed design
4. Plan and perform design review
5. Develop required software
6. Review and test
   6.1. Perform code review
   6.2. Perform isolated software component testing to verify that input-output behavior meets the requirements
   6.3. Perform integration software testing of component within full software system on hardware-in-the-loop
7. Document results, including software issues that must be addressed, and update requirements as needed
8. Iterate until all software requirements are met

Software Release Procedure

1. Assign version control Package Release Number
   1.1. Document version numbers of software sub-components that compose the release
   1.2. Tag the TeamForge or EIMS repositories with the Package Release Number
2. Assemble installation package including release notes. Release notes shall include issue tracking metrics and summaries.
3. Deploy installation package to the hardware-in-the-loop test environment
4. Complete hardware-in-the-loop verification tests
   4.1. When Package Release has passed, obtain signatures documenting approval
   4.2. Archive report and approvals under the Package Release Number
5. Transfer package to SWiFT

Deployment Procedure

1. SWiFT site lead verifies approvals and test report
2. Operators examine release notes
3. Copy release package to network storage
4. Install the release package on each asset
5. Document the date and time of installation on each asset
6. Perform software commissioning checks

3. SOFTWARE DEVELOPMENT PROCESS

Purpose

The purpose of the software development process is to produce a correctly functioning product for the intended application. Generally, the software development process includes design, implementation, and testing of the software products or reuse of existing implementations. Design is the process of defining architecture, components, interfaces, and other characteristics of a system or components. Design activities transform requirements into artifacts that are used for the development of software. These artifacts implement the requirements and are updated to reflect the "as built" product. Design reviews are an important aspect of software development. This process incorporates activities from the following SSQAP practice areas (see Appendix A):

Technical Solution [TS]
Product Integration [PI]
Problem Reporting and Corrective Actions [PR]

Relationships

This process is driven by results of the following process areas:

- requirements engineering
- process implementation and improvement
- project planning, tracking, and oversight
- software verification practice areas

For example, the requirements engineering process results in the software requirements that the software development process must address. The project team creates the required product artifacts (e.g., user documentation, developer's guide, and installation guide) using the documented project processes. The impact of implementation choices on design is continuously incorporated. Relevant stakeholders are informed of issues and included in decisions. Documentation of a design supports development, product maintenance, tracing of requirements, verification, and end users. Issues presented by software verification may require revisiting the software development activities.

Steps

STEP #	STEP DESCRIPTION	STEP ACTIVITY
1	**Analyze requirements, create high level design, and assign tasks**	▪ Analyze requirements applicable to the new functionality ▪ Generate one or more high level preliminary designs ▪ Based upon feasibility, choose a preferred design and document the selection process ▪ Assign requirements to team members based on current work load, expertise and ability to complete the task in the specified timeframe
2	**Identify Software Development Methodology and Implementation Approach**	▪ Evaluate the assigned tasks ▪ Determine the software development methodology used by the development team and identify coding guidelines ▪ Identify the approach for completion of the tasks from a design perspective (e.g., what algorithm must be implemented, how will the specification of the requirement be completed [input], etc.)
3	**Document design**	Capture and document design artifacts:

STEP #	STEP DESCRIPTION	STEP ACTIVITY
		▪ Communicate the design with project team and project stakeholders ▪ Convene meetings with project stakeholders and collaborators to address large design issues or interfaces resulting in documentation
4	**Identify reviewers and perform design review**	▪ Identify reviewers for the design and code ▪ Convene periodic meetings with all developers and relevant stakeholders to reassess requirements tasks and agendas ▪ Discuss problems, delays, and anything that might have slipped through the cracks ▪ Reconsider requirements coverage, tasks, and assignments ▪ Update design documents as necessary to reflect changes made
5	**Develop required software**	▪ Modify the code base in a manner sufficient to capture the required functionality ▪ Comment code at a level suitable for traceability and maintenance ▪ Adhere to project team coding guidelines ▪ Commit all code to version control
6	**Review and unit test**	▪ Review all designs and code ▪ Unit test all code in preparation for verification (formal testing) ▪ Document all reviews and test results ▪ Ensure all action items as a result of reviews and testing are collected in an issue tracking log and are managed to closure
7	**Finalize documentation**	▪ In some cases and sometimes in response to verification results, update manuals. ▪ Ensure that the proper changes to supplemental document artifacts are completed and updated ▪ Document traceability of design, code, and test to requirements

Process Model

Another way of looking at the above process steps is in terms of a process model. The V-model depicted in Figure 1 is one common software development model. The left side represents the sequential execution of development processes: defining requirements, designing a solution, and implementing the solution. The right side represents the sequential execution of testing which validates that the solution has fulfilled the design intent and has met the requirements. The horizontal lines extending from the left side to the right side indicate that testing of the end product is planned in parallel with each corresponding phase of development.

As a result, requirements and design descriptions must be written such that they are testable.

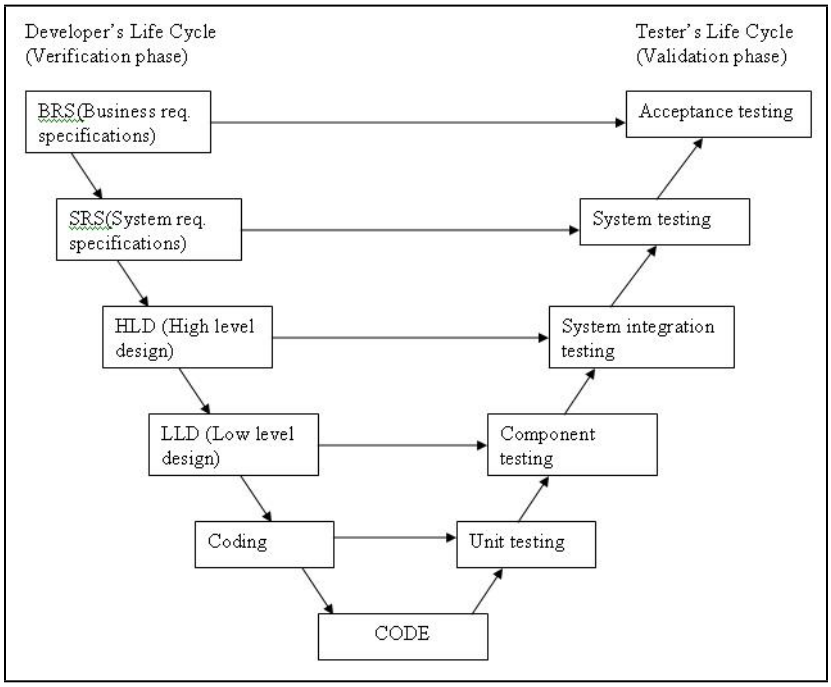


Figure 1. Diagram of the V-model.[2]

The various phases of the V-model are as follows (from Ref [1] with modification):

Requirements like Business Requirement Specifications and System Requirement Specifications begin the life cycle model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering. For SWiFT, business requirements can also be thought of as customer/operator/end-user requirements of how the system is expected to perform.

The high-level design (HLD) phase focuses on system architecture and design. It provides overview of solution, platform, system, product and service/process. If any of the requirements are not feasible, a resolution is found and the requirements are edited accordingly. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

The implementation phase is where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers.

[2] Image credit: http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/

The rigidity of the V-model is both an advantage and disadvantage. It is an advantage if it helps to ensure requirements and testing criteria are well-reasoned and well-documented. But experienced and efficient programmers will recognize that software development is an iterative process – in the sense that early prototypes can help to refine the requirements. So in practice, the SWiFT software development team is encouraged to incorporate the iterative aspects of other software development models, but with the following restrictions:

- The first step of defining requirements must be emphasized and have sufficient time allocated to identify requirements. Developers must not assume requirements will be defined as time passes.
- As requirements are refined through prototyping, the requirement specifications must be updated and approved. Corresponding test criteria must also be updated and approved.
- Iterative developments to the turbine control software shall not be deployed to the turbine hardware until all requirements and test criteria have been fully updated and after the software has been fully validated against the approved requirements. Configuration management will be employed to track what changes have been made and peer-review will determine whether changes have been captured by the requirement specifications.

Although the V-model encourages test planning in parallel with each stage of development, it is also important that the testing phase includes exploratory testing – discovering software responses that were not specifically planned for in the requirements. Thus, the test criteria must be reviewed independently. The test criteria should not focus solely on expected user actions but also unexpected user actions. The tester must not assume the user/operator or external inputs will behave as intended by the software developer.

4. CONFIGURATION MANAGEMENT PROCESS

Purpose

The purpose of configuration management (CM) is to provide a controlled environment for development, production, and support activities. CM is the process of identifying and tracking change. Software product artifacts are identified and managed and baselines of these artifacts are designated. Issues and change requests associated with the software product are tracked. Backup and disaster recovery of software product artifacts are implemented. This process must ensure retrieval of any baselined artifact over the project's lifetime. While multiple projects may share a common CM process and practices, the processes and practices are still performed by each project.

This process incorporates activities from the following SSQAP practice areas:

Configuration Management [CM]
Problem Reporting and Corrective Actions [PR]

Relationships

The configuration management of artifacts and work products is ongoing throughout the lifecycle of the product. CM includes identification of configuration items, version control, baselining, and restoration of baselines. The CM interacts with all other practice areas.

The issue tracking process receives issues from internal and external stakeholders including developers. Issues are identified through all the practice areas and tracked through all applicable states or stages, an archive of issues is maintained, and reports to the end-users and stakeholders are provided as appropriate.

Backup and recovery is performed on artifacts from each of the practices as well as artifacts in intermediate states. Backup on a scheduled basis and recovery on a scheduled and unscheduled basis are provided for baselines, codes, documents, and any other identified configuration items.

Backup and recovery services will be provided by Sandia corporate services. It is the responsibility of the SWiFT software project team to place configuration items into version control in the Corporate TeamForge or Corporate EIMS repositories.

Steps

The following steps are not necessarily sequential and are reiterated on an as-needed basis.

Step #		Step Description	Step Activity
1	**Identify and Collect Software Configuration Items**		
	1a	**Identify and collect artifacts of practices**	▪ Document the artifacts to be placed under configuration ▪ Place configuration items in project team repository

Step #		Step Description	Step Activity
	1b	**Identify and collect other items**	▪ Identify other items that need to be configuration controlled ▪ Collect other items for configuration control
2	**Version Control Software Configuration Items**		
	2a	**Identify team process for dealing with multiple paths of development**	▪ If multiple paths of development will be taken, determine the process for controlling that change (e.g. locking, branch and merge, manually merging individual projects or multiple versions of the same artifact)
	2b	**Determine types of repositories**	▪ Determine the type(s) of repositories needed for each type of configuration item (e.g., Git, cvs, subversion, PVCS Version Manager, WebFileshare, Records Management System, DOORS, shared file system)
	2c	**Create repositories**	▪ Create a repository for each repository type
	2d	**Archive each item**	▪ Determine in which repository each software configuration item belongs ▪ Place each software configuration item in a repository
3	**Baseline and Rollback releases**		
	3a	**Plan for baselines**	▪ Determine what types of releases are needed (e. g., test releases, minor and major releases, emergency releases, documentation releases) and any common schedules ▪ Determine which version control repositories a release draws from ▪ Baseline a set of software configuration items initially for each version control repository ▪ Develop a method of composing release numbers to reflect release decisions (10.2.003 reflects that 10 is a major release with architectural differences from the previous release 9, 2 is a minor release with enhancements, and 003 contains bug fixes). ▪ Use the release numbers in naming baselines ▪ Determine how a promotion model (for example, “dev, qual, prod”) will be supported by baselining ▪ Baselines may also be needed at transition points indicated by the course of the software development or may be scheduled on a periodic basis
	3b	**Generate baseline**	▪ Tag a baseline or release in each participating

Step #		Step Description	Step Activity
		and corresponding releases	repository
	3c	**Rollback releases**	▪ Roll back a release to a previous release when a critical failure occurs in production or when a design path proves to be flawed (the release may be a set of class files delivered, an entire tagged release, or a tagged component of a release depending on the needs of the development team)
4	**Track Issues**		
	4a	**Plan for issue tracking**	▪ Determine the process flow for tracking issues ▪ Determine states for issues (e.g., New, Opened, Approved, Implemented, Verified, Closed)
	4b	**Create a tracking system**	▪ Create a tracking system for issue tracking that implements the process for issues to flow from one state to another state (e.g., bugzilla, PVCS Tracker, Excel spreadsheet)
	4c	**Track issues**	▪ Insert issues in the tracking system ▪ Review new issues and assign properties (e.g., severity, owner) ▪ Track issues as their states change
	4d	**Report**	▪ On a release or test schedule, report metrics (e.g., number of open issues, number of open issues of high severity)
5	**Backup and Restore Archives**		
	5a	**Backup repository archives**	▪ Backup repositories for version control and issue tracking on a predetermined scheduled ▪ Store repository archives periodically in an off-site location
	5b	**Restore archives**	▪ Restore archives on a periodic basis to test the retrieval function ▪ Restore archives on an as-needed basis when items contributing to an archive are corrupted

Procedure for Software Release

1. Assign version control Package Release Number
   1.1. Document version numbers of software sub-components that compose the release
   1.2. Tag the TeamForge or EIMS repositories with the Package Release Number
2. Assemble installation package including release notes. Release notes shall include issue tracking metrics and summaries.
3. Deploy installation package to the hardware-in-the-loop test environment
4. Complete hardware-in-the-loop verification tests
   4.1. When Package Release has passed, obtain signatures documenting approval
   4.2. Archive report and approvals under the Package Release Number
5. Transfer package to SWiFT

List of Git Repositories on SWiFT TeamForge

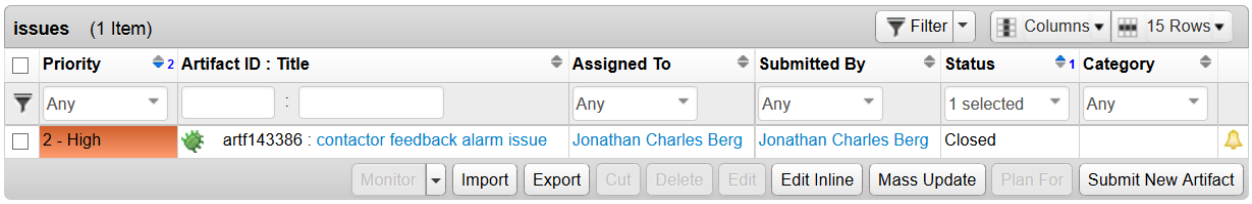
The SWiFT TeamForge site is located at https://teamforge.sandia.gov/sf/projects/swift_facility/ (internal Sandia network access only). A user of TeamForge must have a TeamForge account (request through Sandia WebCARS) and must be given permission to access the SWiFT TeamForge site (granted by SWiFT software lead).

Repository Name	Directory Name	Software in Repository
SWiFT Controller	swift_control	Simulink and Labview code for turbine controller subsystems which is compiled into Veristand models.
SWiFT Software	swift_software	Labview code for the User Interface and Custom Devices as well as Veristand Projects for Turbine Controller and Met Tower
SWiFT Compiled Custom Devices	swift_custom_devices	Compiled Custom Devices reside at C:\Users\Public\Documents\National Instruments\ NI VeriStand 2011\Custom Devices\ on the Veristand Gateway computer. This entire directory is kept under version control in this repository.
SWiFT Hardware-in-the-Loop	swift_hil	All software for hardware-in-the-loop turbine simulation and automated software testing capability.

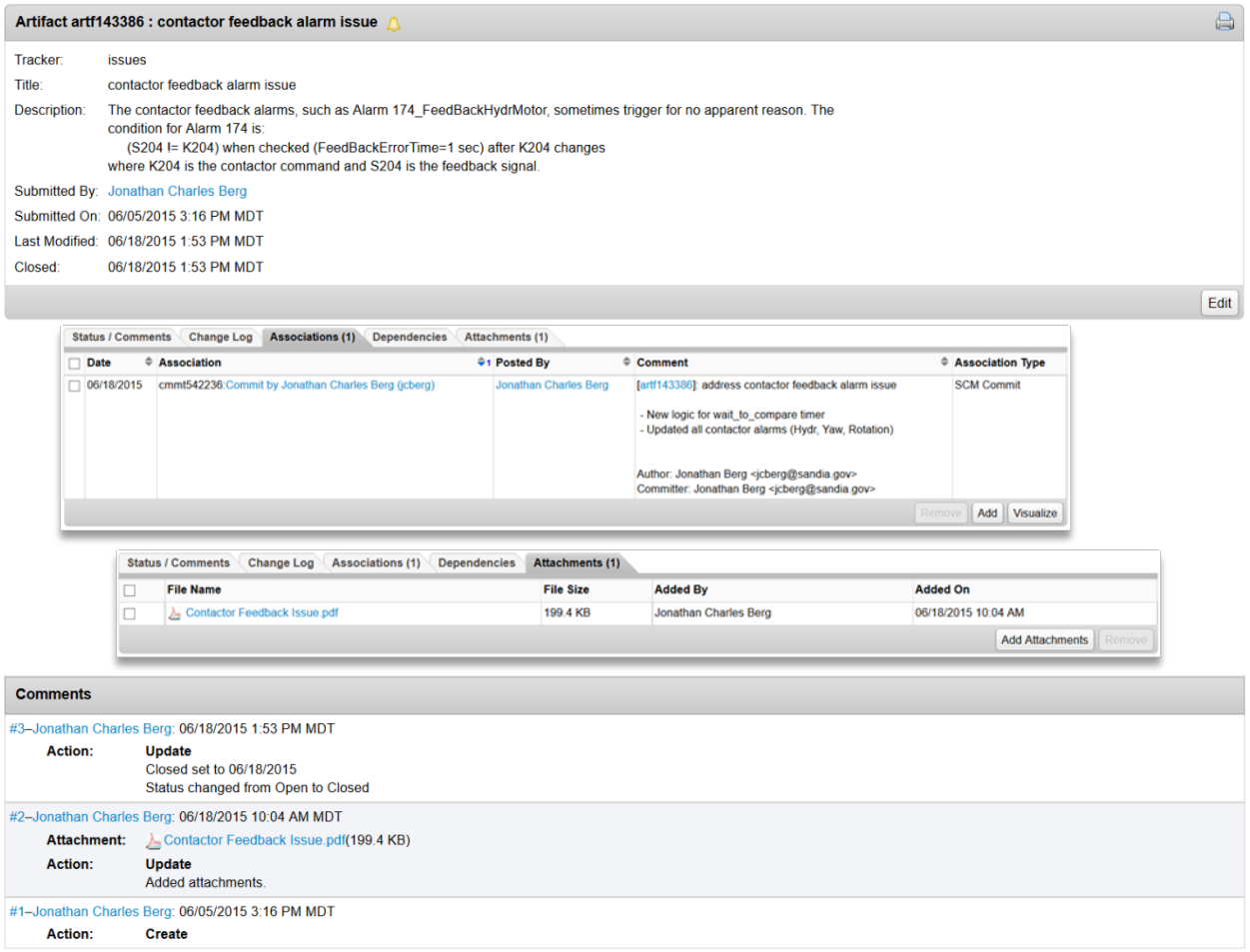
Examples of Git and TeamForge Configuration Management Capabilities

Trackers

Here is an example of the tracker interface which is used for tracking software defects, issues, feature requests, etc.



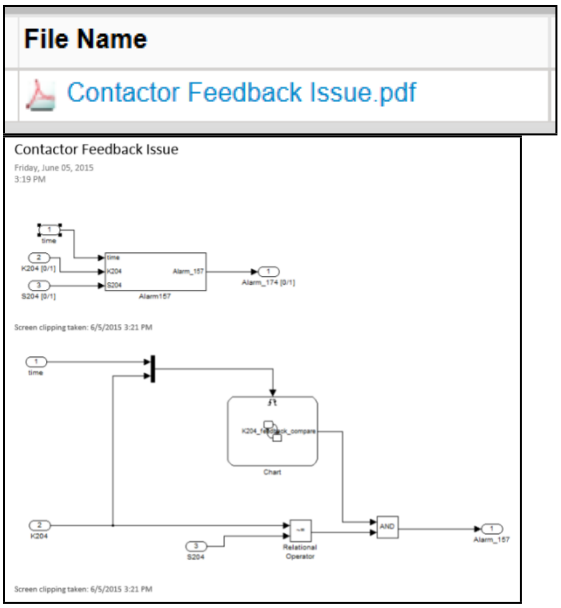
Clicking on the link for "artf143386" opens the View Artifact page:

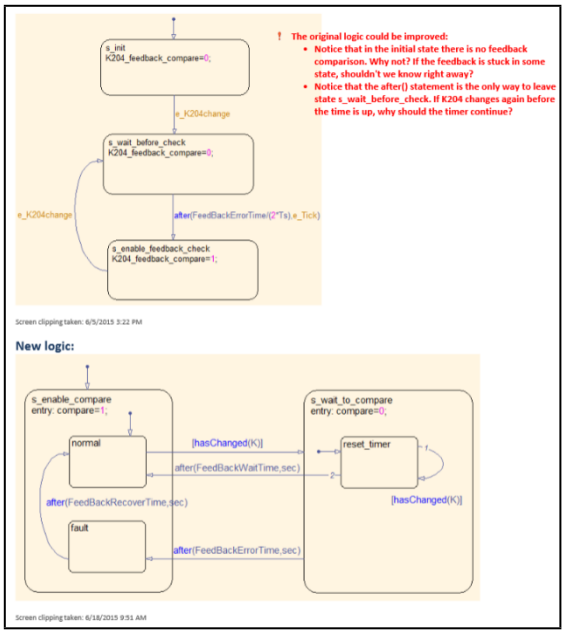


"Associations" allow links to be made between the software repository and the artifact tracker. In the view above, artifact [artf143386] has been linked to software commit [cmmt542236]. The link provides bi-directional tracing of issues and their resolution. This means that someone browsing the software commit history will be able to see and follow the link to this artifact, and

someone browsing the tracker history will be able to see and follow the link to the software commit which fixed the issue.

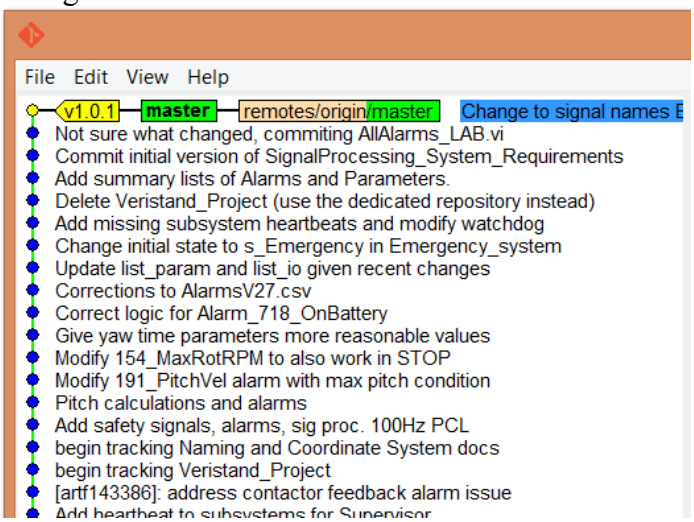
“Attachments” provide a way to add documentation to the tracker artifact. In this case a PDF has been attached which has notes on the issue resolution.





Software Commit Tagging

Below is a visualization of the commit history for “swift_control”. The yellow tag labelled “v1.0.1” is a tagging feature provided by the TeamForge/Git configuration management system which we will use to tag each repository when a software package is pulled together for release. This tagging system allows the various repositories to develop independently and then linked when they are packaged together in a software release.



6. RISK MANAGEMENT PROCESS

Purpose

The purpose of risk management is to identify, address, and mitigate potential sources of risk before they become threats to the successful completion of a project. Risk management spans the lifetime of the project.

Risk management is integral to the planning, tracking, and oversight activities concerning cost, schedule, and performance. The objective of risk management is to increase the probability of project success by controlling threats to program goals in a cost-effective manner.

This process incorporates activities from the following SSQAP practice areas:

Risk Management [RK]

Relationships

This process is driven by results of the requirements management, the project planning, tracking, and oversight, and the software verification practice areas. The results of the requirements management processes provide identification of the subject matter experts who can help with the identification of potential risks. The results of the risk management process will influence how the project is managed and tracked. Most project plans will include a section identifying risk events and how they will be managed. Technical reviews of project artifacts will furnish the basis for monitoring identified risks.

Steps

STEP #	STEP DESCRIPTION	STEP ACTIVITY
1	**Plan the risk management approach**	• Identify risk management requirements and stakeholders • Define the risk database or a system by which risks will be recorded, monitored, and tracked • Identify training requirements associated with defining, recording, monitoring, and tracking project risks
2	**Identify risk events, owners, and agents**	• Identify all known risk events by: ▪ Gathering information for determining the risk events from sources such as issues tracking, peer review results, software quality self-assessments ▪ Consulting existing categorized risk checklists for similar activities • Identify the risk owner(s) and risk agents(s) for each risk event. Owners have responsibility for the risk and risk agents are those affected by the risk • Describe each risk using a common format in the risk database
3	**Analyze and prioritize identified risks**	• Determine the impact severity and likelihood of the identified risk events • Prioritize in descending order using a combination of each risk's severity and likelihood scores or by assigning a ranking based upon expert judgment, risk tolerance, culture, or programmatic

STEP #	STEP DESCRIPTION	STEP ACTIVITY
		need • Group risks of 'common' priority into one of the following response categories: *manage* (proceed to step 4); *watch* (move to a parking lot and revisit occasionally rather than manage actively); *accept* (no action will be taken)
4	**Define the risk responses**	• Plan options that will facilitate opportunities and reduce threats to the project's goals by ▪ Developing a risk response (action) for each risk event ▪ Identifying triggers (symptoms) that indicate the impending threat of a risk event ▪ Determining associated thresholds that indicate when a risk becomes unacceptable ▪ Looking for a common cause that several risks may share thereby allowing control with one response
5	**Monitor identified risks**	• Collect timely, accurate, and relevant information concerning identified risks • Determine effectiveness of response actions, changes in the severity of the impact, and changes in the likelihood of occurrence • Report collected information to the appropriate recipients for determining if there is a problem, whether risk responses have been executed and are progressing as planned
6	**Implement risk actions**	• Execute risk actions as specified in the associated risk response action • Record the current status of the risk response actions (open, closed, deferred, etc.)

7. REQUIREMENTS ENGINEERING PROCESS

Purpose

The purpose of requirements engineering practices is to capture, develop, validate, track, and control the product requirements. Product requirements typically span hardware, software, operations, support, documentation, product training, and other aspects. Requirements are based upon project mission, stakeholders' stated and implied needs and expectations, and organizational commitments. Changes to requirements must be managed throughout the lifetime of the project. Requirements should be reviewed and approved by appropriate stakeholders.

This process incorporates activities from the following SSQAP practice areas:

Requirements Development [RD]
Requirements Management [RM]
Stakeholder Involvement [SI]

Relationships

The project plan provides the initial identification of key stakeholders and the project mission statement as inputs to this process area. Project planning determines whether and when requirements will be implemented. Risk management activities analyze and try to control events that affect the ability to satisfy requirements. A product release identifies requirements that are newly satisfied in that release. Acceptance testing evaluates whether the product meets the specified requirement.

Steps

STEP #	STEP DESCRIPTION	STEP ACTIVITY	STEP OUTPUT
1	**Plan the requirements management (RM) infrastructure**	▪ Identify project stakeholders, organizational commitments, and other sources of requirements. ▪ Identify general or high-level product technical and non-technical requirements ▪ Identify test types ▪ Identify resources ▪ Review project lifecycle model to ensure requirements management is addressed adequately in each phase.	▪ Infrastructure for implementing requirements management ▪ Inclusion of requirements management planning in the Project Plan or a separate Requirements Management Plan
2	**Gather candidate needs, expectations and requirements**	▪ Gather candidate needs, expectations and requirements ▪ Receive change requests ▪ Gather acceptance criteria ▪ Perform initial analysis (e.g., general scope, group and categorize candidate requirements and change requests) ▪ Identify resources needed/availability	▪ List of candidate needs, expectations, requirements, and acceptance criteria ▪ General scope of candidate requirements and change requests ▪ Informal list of resources
3	**Analyze, derive, negotiate and document**	▪ Perform detailed requirements analysis --Revise candidate requirements, change requests	▪ Database of requirements and relationships within

STEP #	STEP DESCRIPTION	STEP ACTIVITY	STEP OUTPUT
		--Derive component requirements --Establish acceptance criteria --Identify test case for each requirement ▪ Determine scope and technical feasibility ▪ Prioritize requirements ▪ Allocate requirements ▪ Document all formal requirements, acceptance criteria, test cases ▪ Document traceability	requirements tracking software (DOORS) ▪ Project software requirements document
4	**Verify**	▪ Verify the requirements document accurately reflects the end user's needs and expectations ▪ Obtain approval of requirements document	▪ Approved requirements document
5	**Baseline**	▪ Baseline the requirements per the approved requirements document according to the project's Configuration Management (CM) process	▪ Requirements document baseline under CM ▪ S/W requirements baselined under CM
6	**Establish change control**	▪ Establish the requirements change control procedures according to CM processes. An automated tool is strongly recommended.	▪ Requirements change control procedures under CM
7	**Manage and track**	▪ Changes to the project can only occur through the change control process. Each change (e.g., change to existing requirement, a new requirement) must proceed from step 2 through step 6 ▪ Changes must be analyzed considering costs, timelines, and personnel as well as technical feasibility and impact on other requirements	▪ Updated change control records ▪ Updated requirements document ▪ Updated requirements baseline

8. SOFTWARE VERIFICATION PROCESS

Purpose

The purpose of software verification is to ensure (1) that specifications are adequate with respect to intended use and (2) that specifications are accurately, correctly, and completely implemented. Software verification also attempts to ensure product characteristics necessary for safe and proper use are addressed.

This process incorporates activities from the following SSQAP practice areas:

Verification [VE]
Validation [VA]
Measurement & Analysis [MA]

Relationships

Developing and maintaining a software verification plan relies on inputs from all other practices that create or update artifacts. This practice updates the software verification plan. Outputs from this practice need to be version controlled.

Steps

STEP #		STEP DESCRIPTION	STEP ACTIVITY	STEP OUTPUT
4	4a	**Prepare for testing**		
		Prepare for specific test	▪ Identify artifact to be tested and the associated test type(s) per the software verification plan ▪ Identify demonstrations of solution-correctness ▪ Check identified artifact for completeness ▪ Ensure required test tools are available ▪ Review acceptance criteria ▪ Establish test schedule and resources ▪ Update Software Verification Plan, as needed	▪ Identified artifacts to test ▪ Code ready for test
	4b	**Prepare test environment**	▪ Analyze test cases and requirements and identify required test environments ▪ Analyze test environments and identity test tools needed for each environment ▪ Install and configure test tools for each test platform	▪ Test environment established
	4c	**Prepare test cases**	▪ Identify and obtain existing test cases ▪ Analyze current set of test cases for completeness, accuracy, and coverage ▪ Analyze test cases against requirements and identify gaps in current test cases	▪ Test cases complete and ready for test

STEP #		STEP DESCRIPTION	STEP ACTIVITY	STEP OUTPUT
			▪ Update test cases with new data ▪ Provide mapping from test cases to requirements ▪ Develop tests to address gaps in coverage	
5		**Conduct testing**	▪ Conduct all tests according to schedules ▪ Version control test results	▪ Preliminary test results
6	**6a**	**Evaluate and report results**		
		Evaluate test results	▪ Evaluate test results -- identify test issues -- determine that acceptance criteria has been met/not met -- determine that previously tested capabilities continue to perform as expected ▪ Determine follow up activities if necessary ▪ Finalize test results	▪ Test report
	6b	**Report test results to appropriate stakeholders**	▪ Report test results to appropriate stakeholders for action. For example: -- Capture and track test changes and/or issues until closed -- Submit test issues for analysis and corrective action -- Proceed to next development phase -- Implement process improvement	▪ Documented communication of reported results to stakeholders ▪ Tracked action items

Hardware-in-the-Loop Test Environment

The process for validating software on the hardware-in-the-loop system will involve testing all of the software system requirements and design specifications. These tests will be automated so that the suite of tests can be applied consistently. It will also be possible to perform exploratory testing by manipulating the turbine simulator manually.

The hardware-in-the-loop system has a set of controller hardware identical to that of the turbine itself. As shown in Figure 2, another piece of hardware which simulates the wind turbine plant model will substitute for the turbine actuators and sensors. The simulator will mimic the turbine behavior at the electrical connection level by reproducing the signal of every wire at the turbine controller input/output (IO). The testing software will mimic user interface actions by linking into the application programming interface. This capability will enable repeatable sequences of simulated button clicks and data entry that a turbine operator might perform.

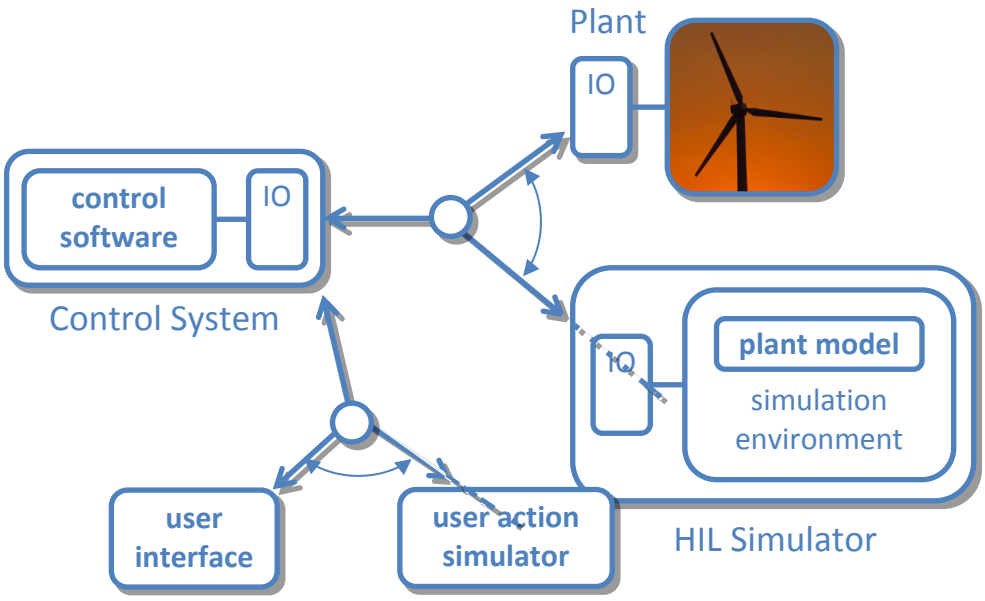


Figure 2: Hardware-in-the-Loop Block Diagram, Ref [2]

Figure 3 shows the connection diagram. Figure 4 and Figure 5 show the completed system.

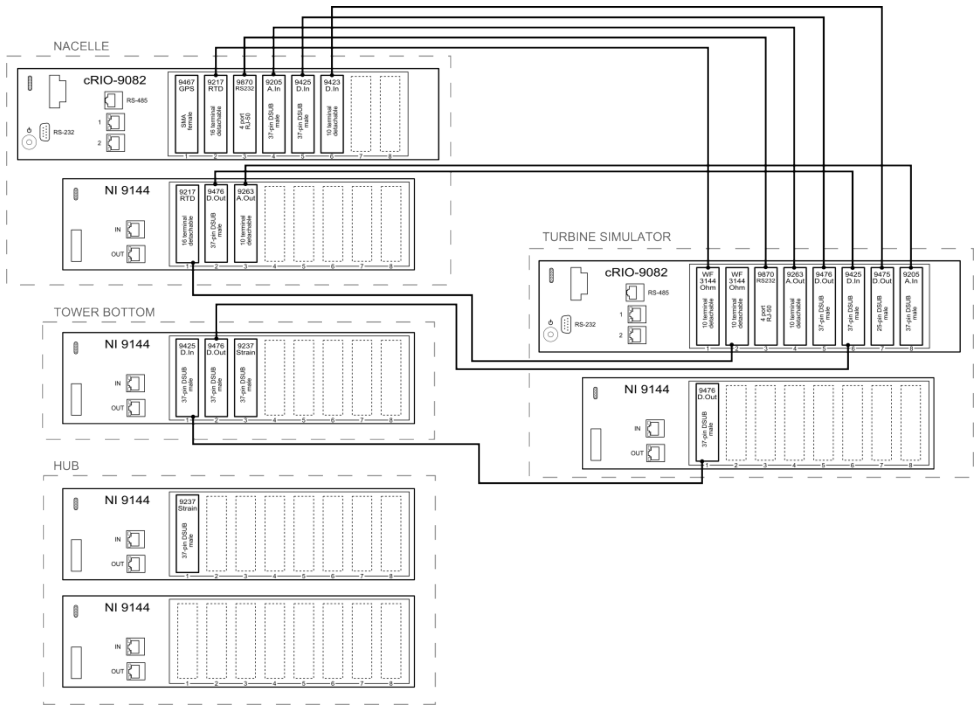


Figure 3: Hardware-in-the-Loop connection diagram.

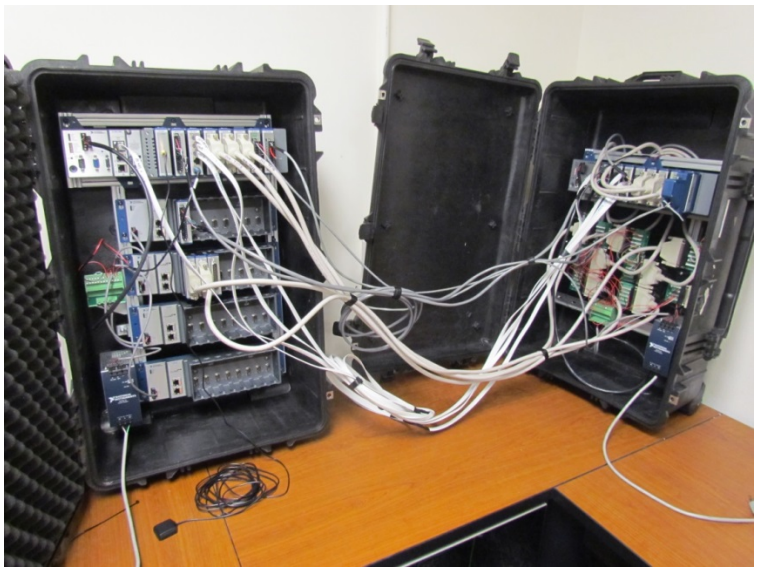


Figure 4: Hardware-in-the-Loop system packaged in two Pelican cases.

Figure 5: Hardware-in-the-Loop drivetrain simulator.

The hardware-in-the-loop system includes a drivetrain with generator and inverter which duplicate (at a scaled size) the behavior and communication of the turbine's generator and inverter. Also included in the drivetrain are a motor and motor drive which will be used to simulate the input torque of the turbine rotor.

9. DEPLOYMENT AND OPERATIONS SUPPORT PROCESS

Purpose

The purpose of deployment and operations support is to assist and train operators in the installation, operation, and ongoing use of the product. Operations support also includes those activities required to manage feedback on the product.

This process incorporates activities from the following SSQAP practice areas:

Deployment [DE]
Life Cycle Support [LS]

Relationships

The operations support plan reflects expectations and requirements for support as specified by SWiFT operators. This plan may influence or be influenced by the project plan. Project issues that are submitted by end-users will be recorded in the project's issue tracking system.

Steps

STEP #	STEP DESCRIPTION	STEP ACTIVITY
1.	**Define and document the Operations Support Plan**	▪ Determine the responsibilities of the team supporting the product. ▪ Identify a point of contact (POC) for questions/issues. ▪ Utilize the Configuration Management issue tracking process for handling issues. ▪ Identify the planned and/or available end-user documentation (theory manuals, tutorials, user manuals, etc.). ▪ Determine the training that the project team will make available to users. ▪ Identify and document product interfaces. ▪ Determine when and how end-user feedback will be solicited. ▪ Document above information as the Operations Support Plan.
2.	**Implement the Operations Support Plan**	▪ Prepare the planned and/or available end-user documentation (theory manuals, tutorials, user manuals, etc.). ▪ Provide assistance for installation, operation, and ongoing use (e.g., schedule, access/download points, documentation, mentoring, etc.). ▪ Provide training for installation, operation and ongoing use (e.g., schedule, tutorials, documentation, classroom materials, self-study materials, mentoring, etc.). ▪ Implement the issues tracking process identified in Step 1 (e.g., submittals, tracking, reports, reviews, etc.). ▪ Solicit operator feedback (e.g., training evaluation

STEP #	STEP DESCRIPTION	STEP ACTIVITY
		forms, verbal feedback, etc.).
3.	**Manage Operator Feedback Regarding Support**	▪ Gather feedback (e.g., issue tracking reports, evaluation forms, emails, verbal comments, etc.). ▪ Evaluate/analyze end-user feedback. - Type of feedback (e.g., compliment, team problem, product problem, etc.). - Degree of satisfaction (e.g., not at all, low, medium, high). ▪ Promote feedback to appropriate project level as necessary to address problems.

Deployment Procedure

1. SWiFT site lead verifies approvals and test report
2. Operators examine release notes
3. Copy release package to network storage
4. Install the release package on each asset
5. Document the date and time of installation on each asset
6. Perform software commissioning checks

10. PROJECT PLANNING, TRACKING, AND OVERSIGHT PROCESS

Purpose

The purpose of project planning, tracking, and oversight is to guide project implementation while balancing, monitoring, and analyzing project quality, cost, schedule, and performance. Project planning includes preparing a plan that describes how the project will be performed and managed. Tracking and oversight include monitoring project activities against the plan and schedule and then, if deviations occur, taking necessary corrective actions to bring projected accomplishments and results back into compliance.

This process incorporates activities from the following SSQAP practice areas:

Project Planning [PP]
Project Monitoring and Control [PO]
Problem Reporting and Corrective Actions [PR]

Relationships

Development of the project plan begins primarily with product information from the strategic plan and the requirements engineering practice area. The project plan may influence or be influenced by negotiated requirements, project processes, operations support, software development, configuration management, software verification, and training. Project issues that impact the project plan (e.g., schedule, costs, performance expectations, etc.) drive modifications and revisions to the project plan.

Steps

Step #		Step Description	Step Activity
1	**Define, and develop the project parameters** (Note: a work breakdown structure or similar tool may provide all that is needed for step 1)		
	1a	**Define the work**	▪ Review project requirements including product expectations and requirements, requirements imposed by the organization and/or end-user, and other requirements that impact the project (such as security). ▪ Identify scope of project in sufficient detail to describe project justification, product, deliverables, and objective estimates of project tasks, responsibilities, and schedule ▪ Identify constraints or limitations (e.g., task duration, resources, inputs, outputs, etc.) ▪ Identify attributes (e.g., size, complexity) of the work products and tasks to estimate effort, costs, schedule, and resources (e.g., labor, machinery, materials, etc.)

Step #		Step Description	Step Activity
			▪ Identify technical approach (e.g., architectural features, established technologies to be applied, breadth of functionality expected in the final products, etc.)
	1b	**Determine major project deliverables**	▪ Define logical decision points at which significant project commitments concerning resources and technical approach are made (e.g., concept exploration, development, production, operations, disposal, etc.) ▪ Identify review schedule for project metrics ▪ Identify major project deliverables that will form the basis of milestones and scheduling
2	**Define the Project Plan (***The following substeps are not necessarily sequential.)*		
	2a	**Plan for data management**	▪ Establish requirements and procedures to ensure privacy and security of the data ▪ Establish a mechanism to archive data and to access archived data ▪ Determine the project data to be identified, collected and distributed
	2b	**Plan for project resources** (e.g., labor, machinery, equipment, materials, and methods)	▪ Identify process requirements ▪ Determine staffing requirements ▪ Determine facilities, equipment and component requirements
	2c	**Plan for needed knowledge and skills**	▪ Identify knowledge and skills needed to perform the project ▪ Assess the knowledge and skills available ▪ Select mechanisms for providing needed knowledge and skills (in-house training, external training, staffing and new hires, external skill acquisition)
	2d	**Plan relevant stakeholder involvement**	▪ Determine relevant stakeholders for each major activity (e.g., those affected by the activity and those with expertise needed to conduct the activity) ▪ Identify roles and responsibilities of stakeholders ▪ Identify relative importance of stakeholder to success of the project ▪ Identify resources (e.g., training, materials, time, funding) needed to ensure stakeholder interaction

Step #		Step Description	Step Activity
			▪ Determine schedule for phasing of stakeholder interaction
	2e	**Plan for significant variances from the Project Plan**	▪ Identify risks (may have been done in the Risk Management practice area) ▪ Identify mitigating responses for those risks (may have been done in the Risk Management practice area) ▪ Determine thresholds for deciding if a variance is significant enough to require attention
3		**Establish the budget**	▪ Determine estimates for effort and cost considering such factors as computer resource needs, externally supplied products, capabilities of tools, facilities needed, level of security, and direct labor/overhead. ▪ Define the budget --committed/expected availability of resources and facilities --incremental funding requirements
4		**Develop the schedule**	▪ Identify major milestones (e.g., event or calendar based) --dependencies between the activities -- appropriate duration of activities -- time phasing of activities --milestones of appropriate time separation ▪ Identify dependencies for various project deliverables that will affect when deliverables can be started or when they need to be completed ▪ Determine due dates for project deliverables based upon estimated effort, available funding, and resources available
5		**Document the Project Plan**	▪ Document the Project Plan including information from steps 1-4.
6		**Review and approve the Project Plan**	▪ Distribute the Project Plan for to management and stakeholders for review ▪ Resolve issues (e.g., lower or defer technical performance requirements, negotiate more resources, find ways to increase productivity, outsource, adjust the staff skill mix, or revise all

Step #	Step Description	Step Activity
		plans that affect the project or schedules) ▪ Update the Project Plan ▪ Obtain approvals from the appropriate management and stakeholders ▪ Obtain commitments from the appropriate stakeholders
7	**Track project performance vs the Project Plan**	▪ Review the Project Plan according to the frequency table. (See Frequency table below.) ▪ Review project metrics per the project metric schedule ▪ Identify significant performance variances from the Project Plan
8	**Implement corrective actions as necessary**	▪ Determine mitigating responses to significant project performance variances ▪ Update the Project Plan ▪ Obtain approvals and commitments

Frequency

For the purposes of this process, frequency refers to the times at which part or all of a process is repeated to address significant variances or changes to the project plan.

Frequency	Description
Initial	The process begins with the documentation of the Project Plan prior to project implementation.
Periodic	Part or all of the process is repeated as necessary for scheduled project milestones.
On-demand	The processes are repeated when events dictate (e.g., product issues, operator feedback, new requirements, discovery of defects in product, etc.). The Project Plan is revisited and revised as events dictate.

11. TRAINING PROCESS

Purpose

The goal of training is to enhance the skills and motivation of a staff that is already educated and trained in the areas of physics, mathematics, and computational science. This practice area is primarily concerned with training that is associated with process implementation. The purpose of training is to develop the skills and knowledge of individuals and teams so they can fulfill their roles and responsibilities. Project teams need to ensure that the training needs of the project are satisfied in accordance with their project plan.

Note: Operator training is addressed in the project's Deployment and Operations Support Process.

Relationships

The training process receives input from the strategic plan and the project plan. However, all phases of the project depend upon the skills and knowledge of the project team to deliver a successful product and satisfy dynamic requirements and expectations. The training process ensures that the team has the required skills.

Steps

Step #	Step Description	Step Activity
1	Perform needs analysis *NOTE: Not all of these activities are the responsibility of the project team but are identified here so that the team is aware that these activities are considered. For example, project management reviews the project plan to capture tasking and resource, but more than likely, it is a team member who identifies a new methodology that could impact the project.*	▪ Review project plan to capture task, resources, roles and responsibilities assignments. ▪ Compare the actual skills and knowledge of team members to the skills and knowledge necessary to fulfill their roles and responsibilities. ▪ Ensure any organizational training needs are identified. ▪ List all gaps between actual skills/knowledge and the skills/knowledge required to fulfill the project roles and responsibilities for each team member as well as the project team as a group. ▪ Identify potential new concepts, methodologies, or approaches within industry that pertains to this project
2	Plan training	• Determine appropriate training solutions for each gap identified in the needs analysis step (e.g., training classes, internal/external SNL interchange meetings, subject matter expert presentations, conferences, seminars, mentoring, self-study, etc.).
3	Implement the training solutions (e.g., attend class, conference, seminar, interchange meeting; self-study, follow mentoring plan, etc)	▪ Individual team member ▪ Project team
4	Maintain records of all training (e.g., TEDS RTC, project Excel spreadsheet, etc.)	▪ Individual team member ▪ Project team

5	Evaluate and review training needs (e.g., class critique, job performance before/after training, impact of subject matter expert interchange on project, etc.)	▪ Individual team member ▪ Project team

12. REFERENCES

1. http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/ , Accessed 2015-04-13
2. C. Kleijn, “Introduction to Hardware-in-the-Loop Simulation”, http://www.hil-simulation.com/images/stories/Documents/Introduction to Hardware-in-the-Loop Simulation.pdf , Accessed 2015-04-13

APPENDIX A: SSQAP REQUIRED PRACTICE ACTIVITIES TABLE

SWiFT Software Quality Plans and Procedures
Satisfying SSQAP P2 and Safety Software Requirements

Practice Level: **P2**
Control Software: **Safety Software**

Work Products:

- **SWiFT Project Plan**
- **SWiFT Software Quality Assurance Plan**
- **SWiFT Software Requirements Documents**
- **SWiFT Software Risk Register**
- **SWiFT Software Test Sequences**
- **SWiFT Software Deployment Training**
- **SWiFT Software Operator Training / Turbine Operating Manual**

Practice Activities Based upon Practice Level Tiers P1 & P2 and Safety Software Considerations
The table below lists the practice activities required by the Sandia quality software implementation document: Specific Use Specification, Sandia Software Quality Assurance Program (SSQAP). Resulting work products for each practice area are identified.

Process Area / Global Practice	P1 & P2 & Safety Software	Work Products
Project Management		
Project Planning [PP]	P1.1 Estimate the scope of the project, estimates of work product and task attributes, budget and schedule, and the project plan P1.2 Define project life cycle P1.3 Determine estimates of effort and cost P1.4 Plan needed acquisitions and suppliers P1.5 Obtain plan commitment P2.1 Plan for project resources, needed knowledge and skills, stakeholder involvement, and data management P2.2 Review plans that affect the project P2.3 Identify project risks P2.4 Reconcile work and resource levels SS.1 Determine regulatory requirements, legal requirements, and standards SS.2 Establish and maintain safety criteria SS.3 Establish a safety organization structure for the project SS.4 Establish a safety plan SS.5 Manage safety-related suppliers' agreements and safety requirements	**SWiFT Software Quality Assurance Plan** **SWiFT Project Plan**
Project Monitoring and Control [PO]	P2.1 Monitor project planning parameters, commitments, data management, and selected supplier processes SS.1 Monitor, analyze, resolve safety software contributions to safety incidents	Weekly team meetings with software team, team lead review with project lead as appropriate.
Risk Management [RK]	P1.1 Determine risk sources and categories P1.2 Define risk parameters P1.3 Establish a risk management strategy P1.4 Identify risks P1.5 Evaluate, categorize, and prioritize risks P1.6 Develop risk mitigation plans P1.7 Implement risk mitigation plans	**SWiFT Software Quality Assurance Plan** **SWiFT Software Risk Register**

Process Area / Global Practice	P1 & P2 & Safety Software	Work Products
	SS.1 Integrate identification/mitigation/ control of safety-specific software risks	
Requirements Management [RM]	P1.1 Obtain an understanding of and commitment to requirements P1.2 Manage requirements changes P2.1 Maintain bidirectional traceability of requirements P2.2 Identify inconsistencies between project work and requirements SS.1 Uniquely identify and manage safety-specific requirements	**SWiFT Software Requirements Documents** Requirements and relationships will be managed in DOORS.
Requirements Development [RD]	P1.1 Elicit needs P1.2 Develop the customer requirements P1.3 Analyze requirements P1.4 Establish product and product component requirements P2.1 Allocate product component requirements P2.2 Identify interface requirements P2.3 Establish operational concepts and scenarios, and a definition of required functionality P2.4 Analyze requirements to achieve balance P2.5 Validate requirements SS.1 Identify and analyze system hazards that may result from software failure SS.2 Determine system/software safety requirements and allocations SS.3 Apply safety principles, collect safety assurance evidence, conduct safety-impact analysis for changes	**SWiFT Software Requirements Documents** Approved by: Major Stakeholders. Sources of requirements are: • Business Requirements • System Requirements • Vestas Technology R&D requirements as documented in controller specifications • Turbine Operator interview • Safety critical requirements The software development team and SWiFT Project Lead will review and identify safety critical requirements that will be subject to formal monitoring and testing.
Stakeholder Involvement [SI]	P2.1 Manage stakeholder involvement P2.2 Manage dependencies P2.3 Resolve coordination issues SS.1 Manage stakeholder safety concerns, dependencies, and coordination	The major Stake Holders for the software project are: • Manager Wind Energy Technologies Dept. • SWiFT Project Lead • Software Development Team Lead Other Stake Holders are: • Turbine Operators • Software Development Team • SWiFT Experiment Team
Measurement & Analysis [MA]	P2.1 Establish measurement objectives P2.2 Specify measures, data collection and storage procedures, and analysis procedures P2.3 Collect measurement data SS.1 Collect safety assurance measures	**SWiFT Software Quality Assurance Plan** **SWiFT Software Test Sequences** Measurement & analysis activities are performed in software verification.
Software Engineering Categories		
Technical Solution [TS]	P1.1 Design the product or product component P2.1 Develop alternative solutions and selection criteria P2.2 Select product component solutions P2.3 Design to the selected product or product component solutions P2.4 Implement the design solution SS.1 Apply safety principles of isolation, independence, incompatibility, inoperability within design architecture SS.2 Integrate system failure mode analysis and fault tree analysis to support design to reduce potential system hazards	**SWiFT Software Quality Assurance Plan** Selected standard industrial control platform and software. • Base controller logic used on commercial Vestas V27 turbine. • Use Matlab Simulink for Control loop development • Use NI Veristand for software development • Compact RIO controller hardware
Product Integration [PI]	P2.1 Determine integration sequence P2.2 Establish the product integration environment, procedures and criteria P2.3 Review interface descriptions for completeness	Document software and hardware

Process Area / Global Practice	P1 & P2 & Safety Software	Work Products
	SS.1 Review system safety integration interfaces SS.2 Ensure that safety analysis covers potential effects of software failure	
Deployment [DE]	P2.1 Assemble product components P2.2 Package and deliver the product or product component P2.3 Transition supplier products SS.1 Incorporate any special handling, marking, build observation processes for the safety aspects	**SWiFT Software Deployment Procedure** **SWiFT Software Deployment Training** **SWiFT Software Operator Training / Turbine Operating Manual** Approved by: Software Development Team Lead The software development procedure will include: • Version control tagging in TeamForge (Git) • Completion of Software Unit Testing • Completion of Hardware-in-the-Loop Testing
Life Cycle Support [LS]	P2.1 Establish a technical data package for each release SS.1 Apply safety principles, collect safety assurance evidence, conduct safety-impact analysis for changes	**SWiFT Software Deployment Procedure** **SWiFT Software Deployment Training** **SWiFT Software Operator Training / Turbine Operating Manual** The software development procedure will include: • Version control tagging in TeamForge (Git) • Completion of Software Unit Testing • Completion of Hardware-in-the-Loop Testing
Configuration Management [CM]	P1.1 Identify configuration items P1.2 Establish a configuration management system P1.3 Create or release baselines P1.4 Control configuration items P1.5 Track change requests P2.1 Establish configuration management records P2.2 Perform configuration audits SS.1 Special controls for the safety-specific components of the software	**SWiFT Software Quality Assurance Plan** **SWiFT Software Deployment Procedure** The software development procedure will include: • Issues and change requests will be tracked using the mechanisms provided by TeamForge. • All software and deployment test records will be stored in the Corporate TeamForge Git Repository. • Software documentations will be version controlled and stored in the Corporate EIMS repository.
Problem Reporting and Corrective Actions [PR]	P2.1 Analyze issues P2.2 Take corrective action P2.3 Manage corrective action SS.1 Separately identify all safety-related problems/correction actions	**SWiFT Software Quality Assurance Plan**
Software Verification and Validation Category		
Verification [VE]	Verification - assurance that the product meets the requirements established for that work product. P1.1 Conduct internal technical review P2.1 Select work products for verification P2.2 Establish the verification environment, procedures, and criteria P2.3 Prepare for peer reviews P2.4 Conduct peer reviews P2.5 Analyze peer review data P2.6 Perform verification P2.7 Analyze verification results SS.1 Develop the verification evidence for the software safety case argument SS.2 Integrate system failure mode analysis and fault tree analysis to support design to reduce potential system hazards	**SWiFT Software Test Sequences** Approved by: SWiFT Software Development Team Lead, SWiFT Project Lead All release software will include: • Formal Code Review of each safety module • Unit testing of each module for major releases. • Hardware-In-the-Loop testing of control system.
Validation [VA]	Validation - demonstrates that the provided product fulfills its intended use. P2.1 Select products for validation P2.2 Establish the validation environment, procedures, and criteria P2.3 Perform validation P2.4 Analyze validation results P2.5 Evaluation selected supplier work products	**SWiFT Software Test Sequences**

Process Area / Global Practice	P1 & P2 & Safety Software	Work Products
	P2.6 Accept the acquired product SS.1 Integrate safety criteria into the selection of products for validation, validation procedures, and validation criteria	
Training Support Category		
Training [TR]	P1.1 Establish the strategic training needs P1.2 Determine which training needs are the responsibility of the organization P1.3 Establish training records P2.1 Deliver training P2.2 Assess training effectiveness SS.1 Determine if system/software safety training activities are needed SS.2 Conduct high priority safety training	Maintain records of personnel training in EIMS

Guidance Level of Formality
Specific Use Specification, Sandia Software Quality Assurance Program (SSQAP)

Elements	Medium Formality (for P2)
Work Products	Work products contain significant detail, including key concepts and are likely in draft form. Work products are identified in the project plan and are stored in a repository available to all project team members.
Reviews	Low formality plus project lead and appropriate management are involved in reviews. Customers are informed of status of reviews. Key concepts of artifacts are reviewed and approved by team members and appropriate management. Review records become work products.
Training	Low formality plus identification of critical skills redundancy (where cross-training results in several team members who are knowledgeable of key areas). Feedback on effectiveness of training experiences is collected. Training records become work products.
Tools	Low formality plus tools of a more specialized nature to address specific tasks (for example, requirements management, collaborative development/support environments). Tools are available to appropriate project members and management and stakeholders.

DISTRIBUTION

1	MS0899	Technical Library	9536 (electronic copy)


Sandia National Laboratories