Technical Letter Report
TLR-RES/DE/REB-2021-05

FAVOR Software Quality Assurance Plan (SQAP)

Date:

June 21, 2021

Prepared in response to Subtask 1.1 of task order 31310020D0005 / 31310020F0103 entitled FAVOR, REAP, and RPV Analysis by:

Andrew Dyszel
NUMARK Associates, Inc.

Terry Dickson
NUMARK Associates, Inc.

NUMARK Project Manager:

Marvin Smith
NUMARK Associates, Inc.

NRC Project Manager:

Patrick Raynaud
Senior Materials Engineer
Component Integrity Branch

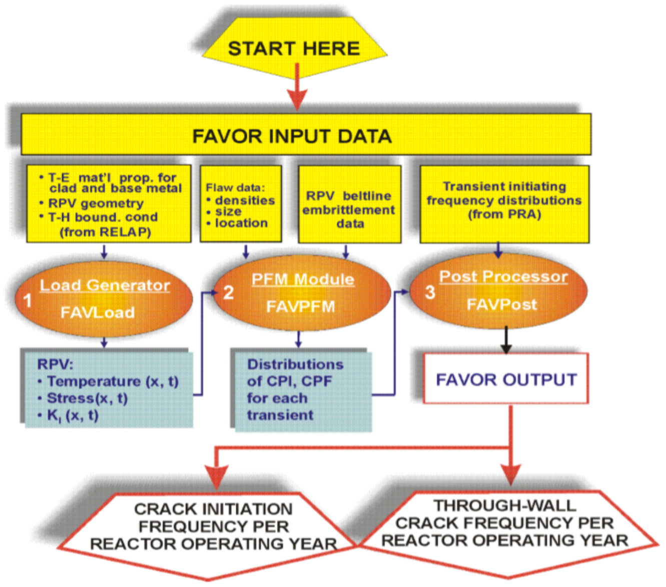
Division of Engineering
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001

Fracture Analysis of Vessels – Oak Ridge (FAVOR)
Software Quality Assurance Plan

Issue Date: 06/21/21

Andrew Dyszel, Terry Dickson, Marvin Smith
NUMARK Inc.

NRC Project Manager: Patrick Raynaud
NRC/RES/DE/CIB


START HERE
FAVOR INPUT DATA
• T-E mat'l prop. for clad and base metal
• RPV geometry
• T-H bound. cond (from RELAP)
Flaw data:
• densities
• size
• location
RPV beltline embrittlement data
Transient initiating frequency distributions (from PRA)
1 Load Generator FAVLoad
2 PFM Module FAVPFM
3 Post Processor FAVPost
RPV:
• Temperature (x, t)
• Stress(x, t)
• K_I (x, t)
Distributions of CPI, CPF for each transient
FAVOR OUTPUT
CRACK INITIATION FREQUENCY PER REACTOR OPERATING YEAR
THROUGH-WALL CRACK FREQUENCY PER REACTOR OPERATING YEAR

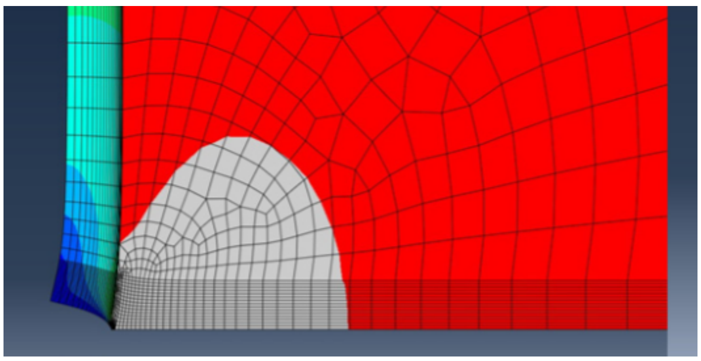


Table of Contents

List of Tables

List of Figures

Project Summary

Project Name	Subtask 1.1: FAVOR Software Quality Assurance Plan
Project Number	Subtask 1.1 of task order 31310020D0005 / 31310020F0103
Internal Project Organization	NRC/RES/DE/CIB

Revision History

Revision	Date	Comments
0	06/21/2021	Original

Approvals

Role	Name	Signature	Date
NRC Project Manager	Patrick Raynaud	Patrick Raynaud	06/29/2021
Lead Software Developer	Terry Dickson	Terry Dickson	06/29/2021
Code Custodian	Patrick Raynaud	Patrick Raynaud	06/29/2021
Software Quality Representative	Andrew Dyszel	Andrew Dyszel	06/29/2021
Contractor PM	Marvin Smith	Marvin Smith	06/29/2021

Acronyms and Abbreviations

This section provides abbreviations and acronyms specific to this plan and software project.

ASME	American Society of Mechanical Engineers
BWR	Boiling Water Reactor
CM	Configuration Management
CMMP	Configuration Management & Maintenance Plan
COTS	Commercial Off-The-Shelf
NRC	United States Nuclear Regulatory Commission
NQA-1	Nuclear Quality Assurance - 1
PM	Project Manager
PMP	Project Management Plan
PWR	Pressurized Water Reactor
QA	Quality Assurance
SDD	Software Design Document
SOW	Statement of Work
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SQE	Software Quality Engineer
SRD	Software Requirements Document
STP	Software Test Plan
STRR	Software Test Results Report

SVVP	Software Verification and Validation Plan
SVVR	Software Verification and Validation Report
V&V	Verification and Validation

Definitions

This section provides definitions specific to this plan and software project.

Assessment	A review, evaluation, inspection, test, check, surveillance, or audit to determine and document whether items, processes, systems, or services meet specified requirements and perform effectively. (NQA-1-2017)
Acceptance Testing	The process of exercising or evaluating a system or system component by manual or automated means to ensure that it satisfies the specific requirements and to identify differences between expected and actual results in the operating environment. (NQA-1)
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 control process. (NQA-1)
Configuration Item	A collection of hardware or software elements treated as unit for the purpose of configuration control. (NQA-1)
Configuration Management (Software)	The process of identifying and defining the configuration items in a system (i.e. software and hardware), controlling the release and change of those items throughout the system's life cycle, and recording and reporting the status of configuration items and change requests. (NQA-1)
Contractor	The organization or organizations contracted by the NRC to work on the FAVOR project.
Error	A condition deviating from an established baseline, including deviations from the current approved computer program and its baseline requirements. (NQA-1)
Graded Approach	The process of ensuring that the level of analysis, documentation, and actions used to comply with a requirement is commensurate with: 1) relative importance to safety, safeguards, and security, 2) magnitude of any hazard involved, 3) the life-cycle stage of a facility or item, 4) programmatic mission of a facility, 5) characteristics of a facility or item, 6) relative importance of radiological and non-radiological hazards, and 7) any other relevant factors (NQA-1)
Independent Reviews/Testing	Person sufficiently independent with respect to the material/product they are reviewing/testing; they did not perform the work they are reviewing or testing. Staff also possess enough subject matter expertise to adequately review/test/evaluate.

Module	A program unit that is discrete and identifiable with respect to compiling; combining with other units, and loading; a logically separable part of a program that can be verified independently and performs a specific limited function, such as modeling physical phenomena, handling user input, output, data storage, etc.; contained, cohesive parts that can be combined to create the final product.
Operating Environment	A collection of software, firmware, and hardware elements that provide for the execution of computer programs. (NQA-1)
Regression Testing	Selective re-testing of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.
Software Design Document	A document that describes the design of a system or component. Typical contents include system or component architecture, control logic, data structures, input/output formats, interface descriptions, theoretical bases, embodied mathematical models, control flow, and subroutines used in the software, and the allowed or prescribed ranges for data inputs and outputs in a manner that can be implemented. Currently described in the FAVOR Theory Manual [1].
Software Design Verification	The process of determining if the product of the software design activity fulfills the software design requirements. (NQA-1)
Software Requirements Document	Documentation of the essential requirements (functional performance, design constraints, and attributes (including acceptance criteria)) of the software and its external interfaces.
Software Verification and Validation Plan (SVVP)	A comprehensive, project-level plan which is a roadmap document that describes the elements, processes, and sequence of actions to ensure that the software properly fulfills its intended use as identified in the Software Requirements Document and Software Design Description Document. These actions may include peer reviews, audits, walkthroughs, analyses, architecture evaluations, simulations, testing, and demonstrations.
Test Case	A set of test inputs, execution conditions, and expected results developed for an objective, such as to exercise a program path or to verify compliance with a specific requirement. (NQA-1)
Test Plan	A document that describes the approach to be followed for testing a system or component. Typical contents identify items to be tested, tasks to be performed, and responsibilities for the testing activities. (NQA-1)
Verification	Mathematical proof of the correctness of algorithms, by confirming that code subroutines and functions produce the expected numerical output.

Validation	The process of evaluating software to determine whether it satisfies specified requirements, by comparing code predictions to experimental data.
Unit Test	Process or code developed to test the numeric accuracy and functionality of new or modified subroutines and functions.
Unit Test Suite	Set of unit tests created while developing and maintaining FAVOR.
Verification Test Suite	Set of input files that exercise all the code options, used to verify that code changes do not negatively impact code performance, and that results are as expected.
Validation Test Suite	Set of input files used to validate the codes' predictions against experimental measurements, to quantify the accuracy, bias, and uncertainty of code predictions.

1 Purpose, Scope, & Applicability

The purpose of this Software Quality Assurance Plan (SQAP) is to define the software quality assurance (SQA) activities that shall be followed during software development, deployment, and maintenance on the FAVOR Codes consisting of FAVLoad, FAVPFM, and FAVPost. This plan defines quality assurance (QA) activities and identifies the documentation that is created and maintained during the entire software engineering process. The goal of the plan is to provide adequate confidence that the software development process is controlled, and that the software products meet established requirements. These activities and their products establish the evidence, credibility, and confidence to ensure that the FAVOR code and its models are adequately accurate and detailed for their intended use.

This plan covers SQA activities for the FAVOR software modules (FAVLoad, FAVPFM, and FAVPost) from development through implementation as defined in both the requirements of the Contractor's Quality Assurance procedure and United States Nuclear Regulatory Commission (NRC) NUREG/BR-0167, *Software Quality Assurance Program and Guidelines* [2].

Scope of work to be performed within the scope of SQA include:

1. Development of new models and subroutines,
2. Update/addition of equations/logic/algorithms,
3. Development and maintenance of input and output visualization tools,
4. Individual model assessment,
5. Integral model prediction assessment,
6. Numerical verification of new or revised code,
7. Validation of new or revised code against data,
8. Maintenance of the FAVLoad, FAVPFM, and FAVPost codes (including correcting code errors, user assistance, and creating new versions and code documentation as needed), and
9. Development of new code assessment cases as new experimental data or analyses with alternative methods become available.

In addition, software development is conducted in accordance with the principles of ASME NQA-1. The version used in establishing this plan NQA-1-2015 [3]. Requirements for software development within NQA-1 have undergone updates to keep with the ever-changing software development environment. Part II, Subpart 2.7 is a subpart that provides specific software requirements.

It is recognized that the FAVOR user community may wish to utilize a later version of NQA-1 in their quality program for the use of FAVOR. It is the responsibility of the organization implementing FAVOR into their QA program to properly perform such an implementation in accordance with all applicable laws and regulations.

2 Reference Documents

The following documents were utilized to develop this plan and/or referenced in this document:

[1] T. L. Dickson, M. L. Smith, A. Dyszel and P. A. C. Raynaud, "TLR-RES/DE/REB-2021-03: Fracture Analysis of Vessels – Oak Ridge FAVOR v20.1.12 Theory and Implementation of Algorithms, Methods, and Correlations," U.S. Nuclear Regulatory Commission, Washington, DC, USA, June 2021.

[2] NUREG/BR-0167: Software Quality Assurance Program and Guidelines (ML012750471), Washington, DC: U.S. Nuclear Regulatory Commission, 1993.

[3] American Society of Mechanical Engineers (ASME), ASME NQA-1-2015: Quality Assurance Requirements for Nuclear Facility Applications, New York, NY: ASME, 2015.

[4] American Society of Mechanical Engineers (ASME), "ASME V&V 10-2006: Guide for Verification and Validation in Computational Solid Mechanics," ASME, New York, NY, December 2006, reaffirmed 2016.

[5] T. L. Dickson, M. L. Smith, A. Dyszel and P. A. C. Raynaud, "TLR-RES/DE/REB-2021-04: Fracture Analysis of Vessels – Oak Ridge FAVOR v20.1.12 User's Guide," U.S. Nuclear Regulatory Commission, Washington, DC, USA, June 2021.

[6] A. Dyszel, T. L. Disckson, M. L. Smith and P. Raynaud, "TLR-RES/DE/CIB-2020-001: Compilation of Software Quality Assurance and Verification and Validation Documentation for the Fracture Analysis of Vessels - Oak Ridge (FAVOR) Software Product (ML20017A171)," U.S. Nuclear Regulatory Commission, Washington, DC, 2020.

[7] A. Ayszel, T. L. Dickson, M. Smith and P. Raynaud, "TLR-RE/DE/CIB-2020-002: Assessment of V&V Efforts of Fracture Analysis of Vessels – Oak Ridge (FAVOR) Software Product – Version 16.1 (ML20017A170)," U.S. Nuclear REgulatory Commission, Washington, DC, USA, 2020.

3 Roles & Responsibilities

The organizational structure and responsibility assignments shall be such that:

- Software development and maintenance is well planned, verified, and documented under quality assurance procedures.
- Quality is achieved and maintained by those who have been assigned responsibility for performing work, and
- Quality achievement is verified by those not directly responsible for performing the work.

Code development is performed by the NRC and/or the Contractor. The NRC is responsible for high level oversight and direction and assigns work based on staffing resources and knowledge.

NRC – the Nuclear Regulatory Commission is managed by Commissioners appointed by the president. The Commission consists of Offices headed by Office Directors. The Office of Nuclear Regulatory Research (RES) is responsible for the FAVOR project.

The significant management level responsibilities associated with NRC organization are provided in NRC management directives.

Contractor – NUMARK Associates, Inc., Managed by a Project Manager. There may be more than one Contractor organization working on the FAVOR project for the NRC, in which case the 'Contractor' refers to all the contractor organizations working on the FAVOR project for the NRC.

The significant management level responsibilities associated with the Contractor are summarized in the Quality Control Manual.

FAVOR Project Organization – The detailed Roles and responsibilities of the project are provided defined in the various sections of this plan and are summarized below. The individual(s) responsible for establishing and executing the project as it relates to SQA activities may delegate work to others but retains responsibility thereof.

NRC Project Manager (PM or COR) is responsible for all aspects of FAVOR work including:

1. Schedules and assigns participants for internal reviews and has internal approval authority.
2. Assigns qualified staff the tasks of performing the responsibilities of code custodian and code developer.
3. Ensures that staff have the necessary training to perform assigned work.
4. Performs reviews of software documentation to ensure completeness and adequacy for the work activity involved.
5. Performs final internal acceptance review of the software product.

Contractor Project Manager (CPM) is responsible for development work done under contract for the NRC, including the management of assigned contract work. Depending on scope of code modifications and NRC direction, a CPM may not be required. If a CPM is required, Responsibilities include:

1. Serves as the technical expert for the overall project/program.
2. Provides QA oversight of the technical content of work activities and deliverables.
3. Raises issues related to unsatisfactory or non-conforming technical performance of project work or deliverables.
4. Controls distribution and revision of procedures.
5. Meets technical and schedule requirements.
6. Assigns qualified staff the task of performing the responsibilities of code developer.
7. Ensures that staff have the necessary training to perform assigned work.

8. Provides software documentation for the quality-related activities that they perform.
9. Performs reviews of software documentation to ensure completeness and adequacy for the work activity involved.
10. Performs final internal acceptance review of the software product.

Code Custodian is responsible for maintaining the electronic files of the code package and preparing and maintaining all documentation in accordance with the CMMP and QA process.

1. Ensures activities comply with the Software Life Cycle and Software Configuration Management.
2. Maintains a strong understanding of how software is installed and operated, how the project utilizes software, the risks associated with using software, and how to utilize the project procedures to manage software configuration.
3. Potentially also acts as an **Independent Technical Reviewer** as defined below.
4. Potentially also serves as **Records Custodian** as defined below.

Software Developer(s) design and implement software code. They resolve problems and verify that all corrections are effective. This includes software updates to the application. Developer responsibilities also include:

1. Performs work activities in accordance with the project procedures and guidance documents.
2. Completes all assigned training in a timely fashion and prior to performing work for the project.
3. Generates various software content (e.g., documents, source code, and QA forms) supporting the quality assurance process.
4. Reviews and/or approves various above software content.
5. Maintains the baseline software and associated files within a secure environment.
6. Manages the controlled software installations.
7. Determines the need for and manage the periodic in-use testing.
8. Initiates the procurement of software related items and services.
9. Performs software testing in accordance with the requirements of the software test plan.
10. Potentially also acts as an **Independent Technical Reviewer** as defined below.

Software Tester(s) create/update all test plans/test cases and provide them to reviewers (as applicable). The Software Tester(s) execute test and document test results. The Software Tester(s) is **independent** from software development tasks they were potentially involved with on this project. They also track all problems/changes requested and resolutions. Tester responsibilities also include:

1. Performs work activities in accordance with the project procedures and guidance documents.
2. Completes all assigned training in a timely fashion and prior to performing work on the Project.

3. Generates various software content (e.g., test plans, documents, and QA forms) supporting the quality assurance process.
4. Reviews and/or approves various above software content.
5. Maintains the baseline software and associated files within a secure environment.
6. Manages the controlled software installations.
7. Determines the need for and manage the periodic in-use testing.
8. Initiates the procurement of software related items and services.
9. Performs software testing in accordance with the requirements of the software test plan; and
10. Potentially also acts as an **Independent Technical Reviewer** as defined below.

Software Quality Engineer (SQE) is dedicated to the verification of compliance with SQA requirements and assisting the FAVOR NRC PM in SQA applications. Responsibilities include:

1. Represents the project on SQA matters.
2. Provides independent SQA inspection, as necessary, to ensure adequate verification coverage.
3. Reviews and approves the SQAP.
4. Reviews quality-related documentation to ensure that SQA requirements and objectives are achieved; and
5. Potentially serves as the **Quality Assurance Manager**, as defined below.

Quality Assurance Manager is responsible for overall project quality compliance. They shall establish an appropriate QA program for the project that meets the applicable requirements. They shall evaluate project performance and QA program effectiveness. They shall serve as the Point-of-Contact (POC) for audits, assessments, and surveillances. They shall perform other project specific duties, such as:

1. Establishes an appropriate QA program meeting applicable requirements.
2. Grades work activities in accordance with the appropriate risk level.
3. Evaluates project QA Program implementation and quality control (QC) activities and performance.
4. Reviews project packages for performance (i.e. meeting acceptance criteria at the required frequencies) and compliance with requirements defined in project documents.
5. Serves as the point-of-contact for audits assessments, and surveillances and ensure the staff has adequate representation during such contacts.
6. Reviews and approves appropriate work-authorizing documents, applicable procedures, and quality training.
7. Reports to project management on the status of the QA program.
8. Assures adequate project training for project staff in QA/QC and in Software QA.
9. Provides independent technical verification of corrective actions taken to address issues reported in Nonconformance Reports and Corrective Action Reports.
10. Performs audits and surveillances as directed by the NRC PM.

11. Coordinates independent audit activities.
12. Identifies quality problems.
13. Initiates, recommends, or provides solutions to quality problems through designated channels; and
14. Ensures that further processing, delivery, installation, or use is controlled until proper disposition of a nonconformance, deficiency or unsatisfactory condition has occurred.

Records Custodian is responsible for maintaining the overall control of documents. The individual:

1. Manages project records, submittals, and record turn over to sponsors.
2. Acts as point-of-contact for all records and documents questions and interactions with staff.
3. Prepares and issues transmittals and communications with the sponsor.
4. Ensures material affecting other areas of the Project Office have been communicated to those individuals responsible for it, such as contracts, procurement, finance, and QA.
5. Adheres to procedures; if not possible, work with the NRC PM in resolving conflicts; and
6. Completes all assigned training in a timely fashion and prior to performing work for the project office.

Independent Technical Reviewer responsibilities include reviewing the technical aspects of a procedure, plan, or deliverable. This may include scientific or technical data recorded in record books, test instructions, and test data packages to ensure data reported within records are of high quality, accurate, traceable, reproducible, and complete. The review responsibilities include the following:

1. Reviewing the technical aspects (technical applicability, correctness, adequacy, and completeness) of a procedure, plan, or deliverables (e.g., letters, topical reports, final research reports).
2. Reviewing the scientific or technical data recorded in a document, test instructions, test data packages, test plans, and benchmarks to ensure data reported within records are of high quality, accurate, traceable, reproducible, and complete.
3. Reviewing calculation packages to ensure they are of high quality, accurate, traceable, reproducible, and complete.
4. Reviewing Software Quality Assurance/Control documents and related forms to ensure they are of high quality, accurate, traceable, reproducible, and complete**.**

A summary of the project team responsibilities are shown in Table 3-1, and a list of key documents that the project team creates during the life cycle of FAVOR development are shown in Table 3-2.

Table 3-1: Functional Responsibility Matrix

P=Prepare/Perform A=Approve I=Input R=Review S=Surveillance OD=Own & Distribute **Documents/Actions**	NRC PM	Contractor PM	Code Custodian	Records Custodian	Software Developer	Software Tester	SQE[1]	QA Manager[1]
FAVOR Software QA Plan (SQAP)	I, R, A	I, A	I	I, OD	I, R	I, R	P, R[3]	I, R, A
Configuration Mgmt. Plan and Procedures (CMMP)	I, R, A	I, A	I	I, OD	I, R		P, R[3]	I, R, A
Software Requirements Document (SRD)	I, R, A	I, R	P, I, R[3]	OD	P, I, R[3]		I, R[3]	S
Software Design Document (SDD)	I, R	I, R, A	I, OD		P			
Source Codes	I, R	I, R, A	I, OD		P			
Acceptance test input files	I, R	I, R, A	I, OD		I, R	P		
Unit & Integration Test Plans[2] (STPs)		A	I, R[3]		I, R	P		
V&V Plan (SVVP)	I, R, A	I, R, A	R[3]	OD	I, R	P	I, R[3]	R, A
Unit & Integration Tests and Results Reports[2] (STRRs)		R, A	I, R[3]	OD	I, R	P		
V&V Tests and Results Reports (SVVR)	R, A	I, R, A	R[3]	OD	I, R	P	S	S

[1] Positions in the Quality Assurance Organization of the Contractor. These positions can be filled by one person, depending on the organization and simplicity of the code change.
[2] Per NUREG/BR-0167, these are classified as informal.
[3] Independent Technical Review

P=Prepare/Perform A=Approve I=Input R=Review S=Surveillance OD=Own & Distribute **Documents/Actions**	NRC PM	Contractor PM	Code Custodian	Records Custodian	Software Developer	Software Tester	SQE[1]	QA Manager[1]
Technical Reviews (e.g., assessments/surveillances)	P, I	P					S	S
Software Changes	R, A	I, R	I, R[3]		P			
Change Documents (Appendices D – L)	R, A	I, R	P, I, R[3]	OD	P		I	S
User Input Guide, Theory Manual	I, R, A	I, R	P, I, R[3]	OD	P, I, R[3]		S	S
Maintaining Problem Reporting, Corrective Action, & Change Control	R, A	R	P	OD	I		S	S
QA Records	A	I, R	R[3]	OD			S	S

Table 3-2: Key Process Documents/Outputs

Process Document/Output
Software Quality Assurance Plan (SQAP)
Configuration Management and Maintenance Plan (CMMP)
Software Requirements Document (SRD)
Software Verification & Validation Plan (SVVP)
Software Verification & Validation Report (SVVR)
Software Design Document (SDD) – may be a part of the FAVOR Theory Manual
Software Test Plan(s) (STPs)
Software Test Results Report(s) (STRRs)
<u>Implementation Documentation</u>   **1. FAVLoad, FAVPFM, FAVPost source code and executables**
2. User's Manual
3. FAVOR Theory Manual
4. Acceptance Test Problems

4 QA & Software Life Cycle Requirements

The FAVOR codes are currently in their operations and maintenance phase within the software life cycle. Consequently, in concurrence with NRC NUREG/BR-0167 [2] and SQA requirements, the following are implemented to ensure quality and integrity of the maintenance and new development of the FAVOR software:

1. Executing the activities and reviews, and preparing the documentation contained within this plan.
2. Evaluating and planning for the development of software (Section 6).
3. Identifying software tools and techniques for the project (Section 14).
4. Identifying and implementing a software development methodology (Section 6).
5. Implementing software requirement activities (Section 7).
6. Implementing software design activities (Section 8).
7. Establishing and implementing configuration management activities (Section 10).
8. Performing software test activities (Sections 9 and 11).
9. Establishing and implementing an issue reporting process (Section 12).
10. Performing periodic reviews of the software and as required by the project and/or Contractor, and/or customer (Section 8.2).
11. Planning for the software release to the customer (Section 17).
12. Maintaining this document to incorporate software changes (Section 1).
13. Maintaining project records related to the software (Section 19).
14. Involving the SQE when required for risk determination analysis and new version/scope change implementation (Section 5.1).

The similarities between the NRC NUREG/BR-0167 Software Life Cycle, the ASME-NQA-1 requirements, and appropriate section of the SQAP are shown in table below.

Table 4-1: NUREG/BR-0167 & ASME NQA-1 QA Software Life Cycle Comparison

NUREG/BR-0167	SQA Work Activity	NQA-1-2015	SQAP Section
NRC Software Life Cycle	Software Life Cycle	Part II, Subpart 2.7 - 101	Section 4
Requirements Definition	Requirements (includes analysis of and defining requirements)	Part II, Subpart 2.7 - 401	Section 7
Design	Design (includes design, coding, integration, and unit testing)	Part II, Subpart 2.7 - 402	Section 8
Implementation	Design Implementation (includes unit and integration testing)	Part II, Subpart 2.7 - 403 and 404	Section 9
Qualification Testing	Test Process (includes internal and customer acceptance testing)	Part II, Subpart 2.7 - 404	Section 9
Installation and Acceptance	Customer Acceptance	Part II, Subpart 2.7 - 404	Section 11
	V&V Testing	Part II, Subpart 2.7 - 402.1	Sections 11.1 & 11.2
Operations and Sustaining Engineering	Problem Reporting, Corrective Action, and Change Control	Part II, Subpart 2.7 – 204,405 and 406	Section 12
	Provide Software Maintenance and Support	Part II, Subpart 2.7 - 405 and 406	Section 18
Retirement and Archiving	Documentation and Records	Part II, Subpart 2.7 – 201 and 404	Sections 17, 19, & 20
Verification and Validation	V&V Testing	Part II, Subpart 2.7 - 402.1	Sections 11.1 & 11.2
Documentation and Deliverables	Documentation and Deliverables	Part II, Subpart 2.7 – 201 and 404	Sections 17 & 19
Project Management	Software Classification	Part II, 200	Section 5
	Planning, Organizing, and Tracking	Part II, Subpart 2.7 - 400	Sections 1, 3, and 6
	Training	Part I, Req 2 -200	Section 13
Configuration Management	Configuration Management	Part II, Subpart 2.7 – 203	Section 10
Nonconformance Reporting and Corrective Action	Problem Reporting, Corrective Action, and Change Control	Part II, Subpart 2.7 – 204,405 and 406	Section 12
Quality Assessment and Improvement	Assessments & Reviews	Part I, Req 18 – 100, Part II, Subpart 2.7 - 203.2 and 403	Sections 8.2 and 16

The FAVOR codes are in the Operation and Sustaining Engineering (OSE) phase of the NRC software life cycle. This correlates to the Operations, Support, and Maintenance phase of the Contractor software life cycle.

Some of the phases in the software life cycle are iterative, although a traditional waterfall software methodology is followed. That is, the major activities within the life cycle traditionally flow from one stage to the next stage without jumping any particular stage or going backwards. The major phases include requirements definition, design, implementation, qualification testing, installation and acceptance, OSE, and finally retirement and archiving. Even though the FAVOR codes are in their OSE phase, when sustaining engineering activities require it, the necessary phases must be revisited accordingly.

5 Software Risk Determination and Description

5.1 Software Risk Determination

Per Contractor SQA requirements for software development, the project is required to follow the detailed Contractor procedures. Per this workflow, this software has been determined to be safety or high assurance software, as the codes are not to be used for making safety decisions. The SQE shall be engaged if scope changes occur that may cause a new risk determination to be performed.

The NRC has defined three levels of software, per NUREG/BR-0167 as:

1. Level 1– Technical application software used in a safety decision by the NRC,
2. Level 2– Technical or non-technical application software not used in a safety decision by the NRC, and
3. Level 3– Technical or non-technical application software not used in a safety decision and having local or limited use by the NRC.

The purpose and use of the FAVOR software are to audit computer codes and analyses developed by the vendors and used by the utilities that are submitted to the NRC for approval. NRC or its licensees may also use FAVOR in their bases for regulatory and/or safety decisions related to vessel integrity evaluations. The NRC has classified the software level as Level 1 for the purposes of this document and future software modifications.

5.2 Software Description

The **F**racture **A**nalysis of **V**essels – **O**ak **R**idge (**FAVOR**) computer program has been developed to perform deterministic and probabilistic risk-informed analyses of the structural integrity of a nuclear reactor pressure vessel (RPV) when subjected to a range of thermal-hydraulic events. The focus of these analyses is on the beltline region of the RPV. Development of FAVOR originated under the

NRC-sponsored Heavy Section Steel Technology (HSST) program and, more recently, continued under the Probabilistic Structural and Material Modeling (ProSaMM) Program, both at Oak Ridge National Laboratory (ORNL).

Thermal-hydraulic events addressed by the FAVOR code include both overcooling accidents and normal operating transients. Overcooling events, where the temperature of the coolant in contact with the inner surface of the RPV wall rapidly decreases with time, produce time-dependent temperature gradients that induce biaxial stress states varying in magnitude through the vessel wall. Near the inner surface and through most of the wall thickness, the stresses are tensile, thus generating Mode I - opening driving forces that can act on possible existing internal surface-breaking or embedded flaws near the wetted inner surface. If the internal pressure of the coolant is sufficiently high, then the combined thermal plus mechanical loading results in a transient condition known as a pressurized-thermal shock (PTS) event. Normal planned reactor operational transients, such as start-up, cool-down, and leak-test can also present challenges to the structural integrity of the RPV.

As shown in Figure 5-1, FAVOR, written in Fortran, is composed of three computational modules: (1) a deterministic load generator (**FAVLoad**), (2) a Monte Carlo PFM module (**FAVPFM**), and (3) a post-processor (**FAVPost**). Also shown are the data streams that flow through the three modules.

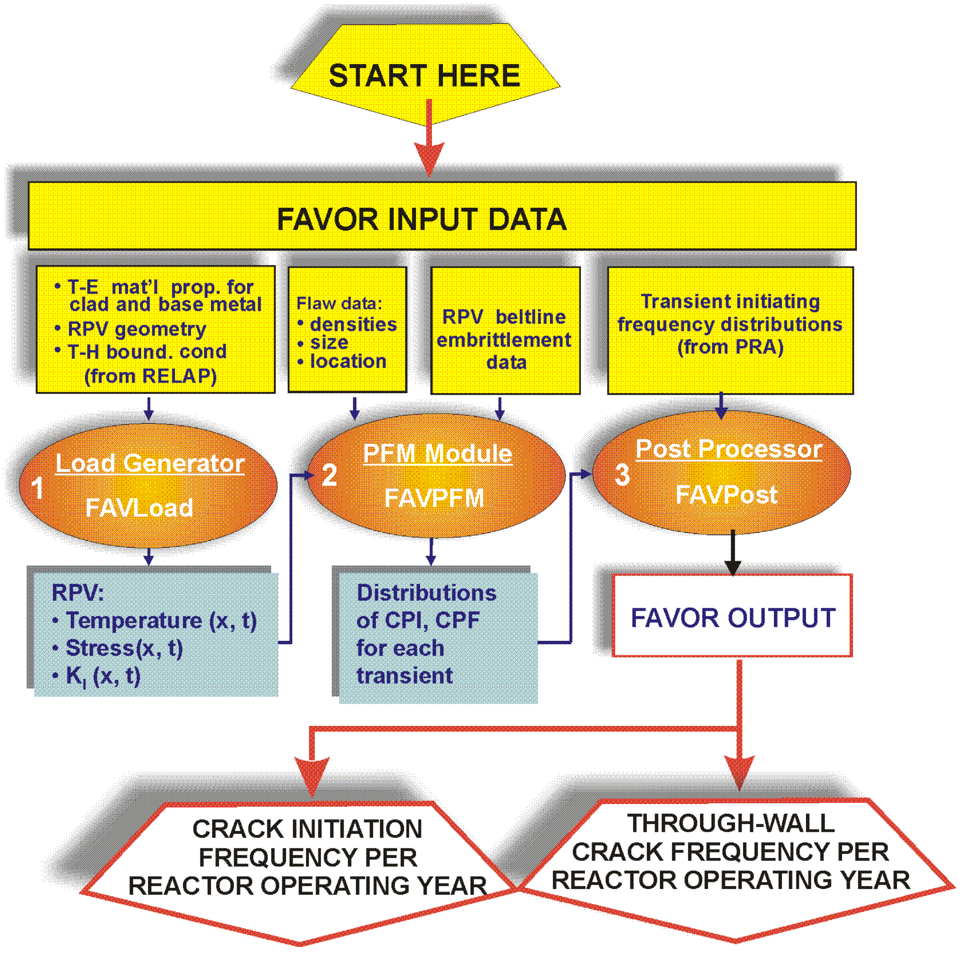

START HERE
FAVOR INPUT DATA
• T-E mat'l prop. for clad and base metal
• RPV geometry
• T-H bound. cond (from RELAP)
Flaw data:
• densities
• size
• location
RPV beltline embrittlement data
Transient initiating frequency distributions (from PRA)
1 Load Generator FAVLoad
2 PFM Module FAVPFM
3 Post Processor FAVPost
RPV:
• Temperature (x, t)
• Stress(x, t)
• K_I (x, t)
Distributions of CPI, CPF for each transient
FAVOR OUTPUT
CRACK INITIATION FREQUENCY PER REACTOR OPERATING YEAR
THROUGH-WALL CRACK FREQUENCY PER REACTOR OPERATING YEAR

Figure 5-1: FAVOR data streams for (1) FAVLoad, (2) FAVPFM, and (3) FAVPost.

The FAVLoad, FAVPFM, and FAVPost codes have been designed to analyze reactor vessels in commercial pressurized-water reactors (PWR) and boiling-water reactors (BWR).

Over the years of development at Oak Ridge National Labs, the focus has been on developing FAVOR to be robust and easy to use and provide the user with an estimate of probabilities of reactor vessel crack initiation and/or failure. Based on [1], prior releases of FAVOR and its predecessors were developed primarily to address the Pressurized Thermal Shock (PTS) issue. Therefore, they were limited to applications involving PWR reactor vessels subjected to cool-down transients with thermal and pressure loading applied to the inner surface of the RPV wall. These earlier versions of FAVOR were applied in the PTS Re-evaluation Project to successfully establish a technical basis to inform the revision of the original PTS Rule (Title 10 of the Code of Federal Regulations, Chapter I, Part 50, Section 50.61, 10CFR50.61). The FAVOR code continued to evolve and to be extensively applied by analysts from the nuclear industry and by regulators at the NRC, to ensure that the structural integrity of aging RPVs is maintained throughout the plant's operational service life including life extension. The v12.1 release of FAVOR represented a significant generalization over previous releases insofar as it included the ability to encompass a broader range of transients (i.e., both heat-up and cool-down) and vessel geometries, including both PWR and BWR RPVs. FAVOR v15.3, included improvements in the consistency and accuracy used for the calculation of K_I for internal surface-breaking flaws. FAVOR, v16.1, includes updates to the flaw-accounting logic in the FAVPFM module and corrections to some cladding influence coefficients for finite internal surface-breaking flaws.

The FAVOR code was subjected to both internal ORNL and external independent verification and validation studies throughout its development lifecycle. At the time of its initial release in 2001, FAVOR was being developed under the Software Quality Assurance (SQA) program at Oak Ridge National Laboratories (ORNL). Subsequent releases of FAVOR were subjected to periodic internal SQA audits; in all cases, the FAVOR code was judged compliant with ORNL SQA procedures and requirements. As the ORNL consensus standard, the ORNL SQA Program is registered to and compliant with the ISO 9001:2008 standard. In 2012, a formal ORNL SQA exemption was granted to FAVOR because the FAVOR software was being developed and maintained with funding from the NRC. The NRC support required that FAVOR be compliant with the terms and conditions of NRC Management Directive 11.7, which requires that all software development, modification, or maintenance follow the general guidance provided in NUREG/BR-0167. ASME Guides and Standards for Verification and Validation (V&V) studies and other references provided more specific guidance (specific to scientific computing applications) during the development of FAVOR. A recent effort to assess the FAVOR SQA against the ASME Code SQA standards [3] and [4] has identified some gaps in the documentation as outlined below. However, NRC has determined that the extensive independent verification and validation studies performed throughout the FAVOR lifecycle provide reasonable assurance that the FAVOR code results are sufficiently accurate and trust-worthy, such that FAVOR may be used to risk-inform regulatory decisions.

Some of the elements of the updated technologies and computational methodology that have been incorporated into FAVOR (from v01.1 to the current release) are as follows:

1. Ability to incorporate new detailed flaw-characterization distributions from NRC research (with Pacific Northwest National Laboratory, PNNL).
2. Ability to incorporate detailed neutron fluence maps.
3. Ability to incorporate warm-prestressing effects into the analysis.
4. Ability to include temperature-dependencies in the thermo-elastic properties of base and cladding.
5. Ability to include crack-face pressure loading for surface-breaking flaws.
6. Addition of a new ductile-fracture model simulating stable and unstable ductile tearing.
7. Addition of a new embrittlement correlation.

8. Ability to include multiple transients in one execution of FAVOR.
9. Ability to include input from the Reactor Vessel Integrity Database, Revision 2, (RVID2) of relevant RPV material properties.
10. Addition of new fracture-toughness models based on extended databases and improved statistical distributions.
11. Addition of a variable failure criterion, i.e., how far must a flaw propagate into the RPV wall for the vessel simulation to be considered as “failed”?
12. Addition of semi-elliptic surface-breaking and embedded-flaw models.
13. Addition of through-wall weld stresses.
14. Addition of base material SIFIC(s) from ASME code, Section XI, Appendix A, Article A-3000, *Method of KI Determination*, for (a) finite semi-elliptical axial and circumferential inside surface flaws and (b) infinite axial and 360° continuous circumferential inside surface flaws into the FAVOR SIFIC database; and
15. Implementation of an improved PFM methodology that incorporates modern PRA procedures for the classification and propagation of input uncertainties and the characterization of output uncertainties as statistical distributions.

A list of key inputs to FAVOR, the important functions and algorithms used in FAVOR, and the FAVOR outputs used in critical decisions are listed in Table 5-1 Some key calculated outputs of FAVOR are K_I (applied stress-intensity factor) time history, through-wall temperature time history, and RT_{NDT} (Reference Nil-Ductility Transition Temperature) at the crack tip. These FAVOR outputs are further used in determining flaw propagation and determining CPI (Conditional Probability of crack Initiation) and CPF (Conditional Probability of Failure).

The GitHub repository allows the testing to be incorporated during the development process by running tests and reporting the success or failure of said tests.

Table 5-1: FAVOR Critical Inputs, Functions, and Outputs

Type	Description
Key Inputs	• Thermo-Mechanical Material Properties for clad and base metal of the RPV (i.e., thermal conductivity, specific heat, density, Young's Elastic Modulus, thermal expansion coefficient, Poisson's ratio) • RPV geometry • Thermal Hydraulic boundary conditions (from RELAP or similar Transient T-H code) • Fast Neutron fluence maps (entered as f_o on Embrittlement Data, described below) • Flaw densities, size, and location (plates, welds, and forgings) • Embrittlement Data (i.e., Cu, Ni, P, Mn, f_o, RT_{NDT0}) • Transient Initiating Frequency distributions (from PRA) • Probability distributions (aleatory and epistemic)
Important Functions and Algorithms	• FAVLoad Deterministic analyses ○ Thermal analysis ○ Stress analysis ○ Linear-Elastic Fracture Mechanics (LEFM) ○ Handling of residual stresses in welds ○ Handling of crack-face pressure for surface breaking flaws • Calculation of Nil-Ductility Transition Temperature, RT_{NDT} • Radiation embrittlement correlations • Fast neutron fluence attenuation and sampling • Handling of K_{IC} and K_{Ia} Databases and calculations of K_{IC} and K_{Ia} • Sampling of RT_{NDT} and RT_{Arrest} • Sampling of Material Chemistry • Flaw characterizations and uncertainty • FAVPFM algorithms and models ○ Warm prestressing logic ○ Truncation for probability distributions ○ Conditional Probability of Initiation (CPI) and Failure (CPF) ○ Post initiation of flaw geometries and orientation ○ Ductile tearing models ○ Initiation-Growth-Arrest (IGA) model • FAVPost algorithm using FAVPFM distributions of conditional probabilities of initiation and failure with input transient initiating frequencies to create fracture and failure frequencies
Critical Outputs	• Temperature as a function of time throughout vessel wall location • Stress as a function of time throughout vessel wall (circumferential and axial) • K_I as a function of time throughout vessel wall • Probability distributions of crack initiation and vessel failure • Crack initiation frequency per reactor operating year • Through-wall crack frequency per reactor operating year

6 Code Development Planning and Assignment

A Software Requirements Document (SRD) shall be generated to define the requirements based upon the current and desired attributes of the FAVOR code. The SRD is a living document that responds to the needs of the NRC and as such additional items may be added to the development process. The SRD typically lists "Additional Items as Needed" under the features to be added, code modification requests, and bug fixes sections. All development planning activities shall be performed under the FAVOR QA processes.

This section provides a description on how and where software planning is implemented and documented. This information shall be documented in the SRD, and in a tracking table (for all tasks) accessible by the NRC PM, the CPM (if applicable), the code custodian, the code developers, and the code testers. Assignments and schedule, as well as major software milestones down to the task levels, shall be documented in the tracking table.

Successful planning needs to consider the project planning aspect of the software development activities. The following items are considered for software planning purposes:

1. Identify the customer and/or customer advocate.
   - NRC
2. Identify customer specific procedures, specific plans, or standards that need to be applied to the project:
   - ASME NQA-1 Quality Assurance Requirements for Nuclear Facility Applications, and
   - NRC NUREG/BR-0167 Software Quality Assurance Program and Guidelines, February 1993.
3. Define and document the organizational structure (Section 3).
4. If software must be procured, follow the procurement workflow (Section 14).
5. If 3rd party licensing is needed, describe all 3rd party software. Items to consider include:
   - Review software licenses for suitability within the software product.
   - Evaluate the risk of using the 3rd party software, including the potential for the publisher to drop support for the software and the potential for changes to the software license.
   - Discuss issues or concerns related to 3rd party software with IP, Legal, Export Control, the customer, and others at the discretion of the NRC PM and/or the CPM.
   - Notify the NRC PM and CPM (if applicable) of the intent to use 3rd party software.
6. Establish a software development methodology.
7. Establish configuration management (CM) (Section 10).
8. Plan for software support and maintenance (Section 18).

Code development planning and assignment should follow a common procedure for FAVOR (FAVLoad, FAVPFM, and FAVPost modules). A widely accepted software control tool, Git, will be used to Capture planning actions and assignments, and the official FAVOR repository shall be hosted on NRC's GitHub enterprise account: www.github.com/NRC-Research.

If SharePoint is being used for any development activities, developers should use the 'check-out' and 'check-in' functions on the SharePoint site when accessing documents or files. The structure of the tasks within the GitHub repository shall be decided and changed as needed by the NRC PM to allow for the following workflow functions:

1. Propose and describe a change to the code.
2. Explain the purpose of the change.
3. Assign a code change task to a developer.
4. Assign a start date and a due date to a code change task.
5. Document the code change task completion date.
6. Document the Git commit corresponding to the assigned change in the source control tool.
7. Document whether the code change is to be a 'developer' option or a 'user' option for the code; and
8. Document what SRD requirement corresponds to the code change. It is important to note that the SRD may need to be updated to reflect the additional requirements resulting in the changes to the code, as applicable (bug fixes may not require changes to the SRD, but new features likely will).

All code development or maintenance work assigned to the Contractor shall be related to a requirement in the SRD or an issue in GitHub, and shall be tracked via GitHub. No code changes shall be performed without being proposed and approved.

7 Software Requirements

This Section provides a description of how software features/requirements shall be implemented and documented. This process consists of documenting, analyzing, tracing, prioritizing, and agreeing (approval) on customer-desired outcomes (requirements/specification). Activities defined should include controlling requirements changes and communication to relevant stakeholders.

The software requirements are usually specified in a Software Requirements Document (SRD). This document provides a description of the requirements that are expected by the customer, i.e., NRC. This includes both the functional requirements of the FAVOR code as well as performance requirements, including individual model requirements and testing. The requirements document also describes the external attributes of the modeling task and feeds into the FAVOR Theory Manual and User Guide. The design document (discussed below) describes the internal attributes of the modeling task and feeds into the programmer's manual. Sometimes the software requirements document and the design document are combined.

FAVOR Software requirements are delineated into three categories: Inputs, important functions/algorithms, and critical outputs.

FAVLoad, FAVPFM, FAVPost shall adequately read and represent the following inputs from the user and process them through intermediate data flows to support the various models within the three modules. These inputs are based on the beltline region of a reactor vessel. Figure 7-1 illustrates a PWR example of the beltline region.

Inputs include the following:

1. Thermo-Mechanical Material Properties for clad and base metal of the RPV (i.e., thermal conductivity, specific heat, density, Young’s Elastic Modulus, thermal expansion coefficient, Poisson’s ratio).
2. RPV geometry (e.g., radius, clad thickness, base metal wall thickness).
3. Thermal Hydraulic boundary conditions (e.g., convective heat transfer coefficient, coolant temperature, and pressure versus time – typically from RELAP or similar Transient T-H code).
4. Fast Neutron fluence maps (entered as f_o on Embrittlement Data, described below).
5. Flaw densities, size, and location (plates, welds, and forgings).
6. Embrittlement Data (i.e., Cu, Ni, P, Mn, f_o, RT_{NDT0}).
7. Transient Initiating Frequency distributions (from probabilistic risk analyses (PRA)); and
8. Probability distributions (aleatory and epistemic).

FAVLoad, FAVPFM, FAVPost shall adequately perform the following algorithms / calculations based on the established models from the user input:

1. FAVLoad Deterministic analyses for the reactor vessel including clad, such as:
    a. Thermal analysis,
    b. Stress analysis,
    c. Linear-Elastic Fracture Mechanics (LEFM),
    d. Handling of residual stresses in welds, and
    e. Handling of crack-face pressure for surface breaking flaws.
2. Calculation of Nil-Ductility Transition Temperature, RT_{NDT}.
3. Use of Radiation embrittlement correlations for irradiated RT_{NDT}.
4. Determination of fast neutron fluence attenuation and sampling.
5. Handling K_{IC} and K_{Ia} Databases and calculations of K_{IC} and K_{Ia}.
6. Sampling RT_{NDT} and RT_{Arrest}.
7. Sampling Material Chemistry for Cu, Ni, P, and Mn.
8. Handling flaw characterizations and uncertainty.
9. FAVPFM algorithms and models, such as:

a. Warm prestressing logic,

b. Truncation for probability distributions,

c. Conditional Probability of Initiation (CPI) and Failure (CPF),

d. Post initiation flaw geometries and orientation,

e. Ductile tearing models, and Initiation-Growth-Arrest (IGA) model

The codes shall appropriately calculate the following critical outputs:

1. Temperature as a function of time throughout vessel wall location.
2. Stress as a function of time throughout vessel wall (circumferential and axial).
3. K_I as a function of time throughout vessel wall.
4. Probability distributions of crack initiation (CPI) and vessel failure (CPF).
5. Crack initiation frequency per reactor operating year; and
6. Through-wall crack frequency (TWCF) per reactor operating year.

New software requirements shall be identified in a Software Requirements Document and shown to be implemented in the software in the Software Design Document in accordance with a software release process. When releasing software, the transmittal shall contain documentation of the test case(s) developed to document the implementation of the requirement(s). These test cases shall be maintained in a broader test suite and may not be part of the routine test suite employed for verification and assessment purposes. Future FAVOR code releases shall follow the requirements of the software release process in a new software requirements document. New software requirements shall be implemented and documented in the following manner:

- Software requirements address the basic issues of functionality, external interfaces, performance, attributes, and design constraints imposed on implementation.
- Each requirement is to be uniquely identified and defined such that its achievement is capable of being objectively verified and validated.
- Each requirement must be testable and traced to no less than one test.

GitHub shall be used to establish the software requirements based on unit tests and integration tests performed for the FAVOR code(s). Due to the detailed nature unit tests, only key tests relevant to a typical FAVOR user shall be used to establish the software requirements. These software requirements shall be described and updated on an ongoing basis in the FAVOR Theory and User's Code manuals, as applicable.

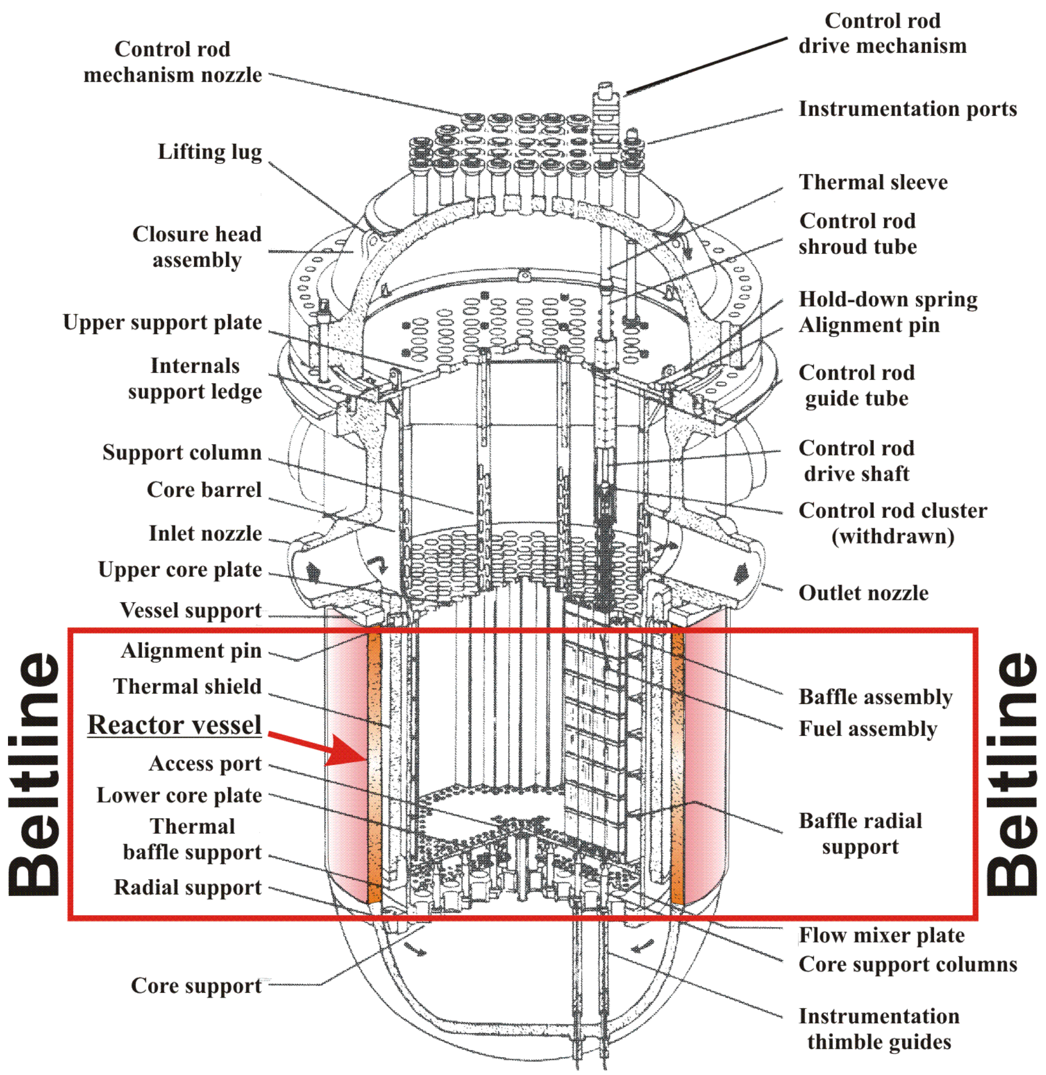


Figure 7-1: The beltline region of the reactor pressure vessel wall extends from approximately one foot above the active reactor core to one foot below the core for a pressurized water reactor (PWR).

7.1 Software Requirements Document (SRD)

The SRD or SRD section for a given module shall identify the function, user input requirements, other data sources, output requirements, interfaces (including any user interactions the software will require), performance requirements, installation considerations, design inputs, and any design constraints for a given module, including operating system, or portability between multiple platforms, as applicable. Applicable references, specifications, codes, standards, regulations, procedures, or instructions shall be identified that establish software requirements, test, inspection, and acceptance criteria. Acceptance criteria may include a quantification of specified acceptable error range per percent, or a quantitative basis for each required output or feature to be evaluated.

The SRD shall also specify technical and software engineering requirements, including safety and security features (e.g., vulnerability protection, and cyber-security) for the FAVOR deliverable product. Security requirements shall be specified commensurate with the risk from unauthorized access or use and shall address any controls of proprietary data (e.g., input data).

Requirements shall be complete, correct, concise, consistent, unambiguous, understandable, relevant, and testable. They shall be uniquely identified using an appropriate and consistent numbering / lettering scheme, such as, identification as a requirements (R), the module or framework acronym, and a sequential number. As stated above, each unit, verification, and validation test documented on the FAVOR repository on GitHub is a de-facto requirement. This allows for the requirements to be demonstrably satisfied in a continuous manner, in line with the principles of continuous integration and Agile software development.

SRDs shall undergo software requirements verification by a technical reviewer(s) not involved in the development of the document. SRDs shall be reviewed by subject matter experts, technical reviewers, and the SQE. SRDs shall be placed under configuration management in accordance with Section 10. The Software Requirements Description Criteria Form FAVOR-SQA-3 (see Appendix C: Software Requirements Description Criteria) may be used as an aide in developing SRDs.

7.2 Functional Requirements

The model development task should meet the expectations that are laid out in the SRD. This document provides a development of the model to be implemented into FAVOR, starting with a description of the physics, possible suitable equations that describe the physics, and simplifications to those equations, and possible correlations to be used. Furthermore, any numerical solution methods used are described along with known limitations to those methods. This description is written in a manner to be readily included into the FAVOR Theory Manual. The requirements document may refer to whitepaper studies that were performed as a proof-of-concept or as a comparison of alternate modeling approaches.

All parameters used in the modeling are described, and parameters that can significantly impact functional performance are highlighted (e.g., number of RPV simulations). All FAVOR input requirements are described along with format for inclusion in the FAVOR User's guide. This should include a description of the acceptable range of each parameter. This document should also include a list of all internal tests and error/warning messages to be provided. Links or comments in the code should be provided to link to the requirements documentation.

The functional requirements description provides the basis for the following documents:

- FAVOR Theory Manual
- FAVOR User's Guide

7.3 Performance Requirements

For all new models, FAVOR's performance should be assessed. Typically, calculation time, number of processors or RAM needed, and/or number of parallel simulations are used as measures of code performance. The developer should perform testing with actual plant baseline runs (e.g., Palisades) to provide meaningful estimates of CPU expectations.

8 Code Design

Due to the history of FAVLoad, FAVPFM, and FAVPost, existing software design is referenced from the code description documents. FAVLoad, FAVPFM, and FAVPost are written in Fortran and should remain that way for new development.

New design shall be implemented as follows:

- The FAVLoad, FAVPFM, and FAVPost codes are written in Fortran and compiled for execution on a Microsoft® Windows PC, Linux, and macOS.
- The Software Requirements Document shall detail the requirements for the new FAVOR software version being released.
- The Software Design Document details how the software shall be structured to satisfy the software requirements.
- Instructions for users to run the FAVOR executables C are included with the release of the code. Note that the source code will not be normally released to users.
- An input generator is distributed as a Microsoft® Excel file.
- Separate source codes and executables (FAVLoad, FAVPFM, FAVPost) are produced.
- Executables shall be distributed with each release of the code.
- New design features are described in the next revision of the FAVOR Theory and or User's Manual, as applicable.

8.1 Software Design Document (SDD)

The SDD shall define the computational sequence necessary to meet the software requirements. The FAVOR Theory Manual [1] currently contains the SDD, but a standalone SDD should also exist. The documentation shall include, as applicable, software architecture, numerical methods, mathematical models, physical models, control flow, control logic, data model, data flow, process flow, data structures, process structures, and the applicable relationships between data structures and process structures. The design of the user interface and design of interfaces with other software shall also be specified. The software design shall consider the computer program's operating environment. Measures to mitigate the consequences of problems, as identified through analysis, shall be an integral part of the design. These potential problems include external and internal abnormal conditions and events that can affect the computer program critical outputs or functionality.

The design documentation shall contain enough information so that the design can be passed to a competent programmer for implementation.

The Code Developer or Code Custodian shall revise the SDD to reflect the new design. The Software Design Description Criteria Form FAVOR-SQA-5 (see Appendix E: Software Design Description Criteria) may be used as an aide in developing SDDs. The design may reveal the need for modification of the associated SRD.

8.2 Reviews

The Software Requirements Document constitutes the basis for Software Design Document. The documentation related to software design is updated as part of the development and release process. Between software releases and other scheduled design reviews, the software design shall be reviewed and approved at the NRC PM's discretion. SDDs shall be reviewed by appropriate code developer, code custodian, CPM (if applicable), and NRC PM. SDDs shall be placed under configuration management in accordance with Section 10. In addition, reviews may include any supporting code documentation and assessment documents, the QA documents, and the code.

9 Code Development and Testing

This section provides a description of how FAVOR code development is implemented and documented. Coding standards, naming conventions, unit testing, and integration testing also is addressed here.

9.1 Coding Standards

The original FAVOR codes were written in Fortran 77/90/95 which has since been superseded by more modern Fortran standards. All new subroutines/functions and code modifications should be written using the standards according to FORTRAN-95 and later Standards and the latest requirements in the ASME-NQA-1 [3].

9.2 Unit Testing

Unit testing shall be performed on new subroutines/functions that are added to ensure that information is properly calculated. Unit testing shall be documented in the FAVOR repository on GitHub. The Software developer and/or software tester designs an appropriate unit test and documents the results of the testing for inclusion in the V&V section of the technical basis document. Any modification made to existing subroutines shall require the developer or tester to ensure that the existing unit tests are adequate and, if not, to develop additional unit tests corresponding to the modifications made.

Unit testing shall be performed to ensure the following:

- The numerical solution is properly being solved (i.e., numerical verification),

- The code is continuous within the range of possible input conditions, and
- All new functionality is properly working.

All the existing unit tests must be run and documented on GitHub. The repository shall be configured with Continuous Integration featured such that tests are automatically run when changes are pushed to the central repository on GitHub. A summary of the testing performed shall be documented in Software Test Reports before releasing a new version of FAVLoad, FAVPFM, and/or FAVPost. All unit tests developed and/or modified as part of the code modification must be submitted along with the FAVLoad, FAVPFM, and/or FAVPost STRs to a second developer for verification purposes. A representative unit test shall be added to the existing unit test suite for continued use.

9.3 Integration Testing

Following unit testing, Integration testing shall be performed on a collection of related units to ensure that functional requirements are being met. For example, a change in FAVLoad subroutine that calculates hoop stress would be verified in Unit testing, but also should be tested with a FAVPFM run to ensure that performance is satisfactory and that no unintentional changes to key outputs such as CPI and CPF are generated. Similarly to unit tests, integration tests shall be part of the GitHub Continuous Integration and thus be documented on GitHub. A summary of the testing performed shall be documented Software Test Reports before releasing a new version of FAVLoad, FAVPFM, and/or FAVPost.

As a special note, NUREG-BR-0167 [2] classifies Unit Testing and Integration Testing as informal testing because a formal test plan is not required, but Unit and Integration testing should still be documented in the FAVOR repository on GitHub and in the applicable STRs.

10 Configuration Management

Software configuration management (CM) shall be established to ensure the following four required outcomes are met:

1. Product versions/baselines can be uniquely identified.
2. Specific versions of deliverables can be reproduced (software, data, and information product deliverables).
3. Unintended and/or conflicting changes are prevented.
4. Unintended use is prevented.

This includes identifying:

1. Processes and tools that will be used for CM of software and software documentation.
2. Configuration items (software and documentation).
3. Control of configuration items.

4. How to track & control changes (Section 10.2).

For more details on the implementation of CM for this project, please refer to the FAVLoad, FAVPFM, FAVPost *Configuration Management & Maintenance Plan* (CMMP). In addition, Section 14 provides various tools and techniques that are used to maintain the various configuration management documents.

10.1 Code Control

Git is used for CM of the FAVLoad, FAVPFM, FAVPost codes. The code custodian will provide developers with the software required to access the Git repository on GitHUB.com. Git is a free and open source distributed version control system. Each software developer shall have a unique username to allow for easy identification from other developers. This program electronically logs all the changes from the previous baseline version(s). Git has the capability to revert to any previous commit made after the repository initialization, maintaining the ability to replicate the source code from any commit. Git creates a unique code identification corresponding to the changes made to allow the developers to identify subversions between releases.

Each new major version of FAVOR is established as a new baseline version in Git. The future changes shall only be tracked off the major version. The previous version shall continue to be maintained on local machines after a major release. Approval for code changes shall be obtained through submittal of a pull request. The form contains the following:

- Reason for change and impacts,
- Description of change, and
- Projected start/end dates.

Unless warranted, the commit made to the repository on GitHub shall be for the proposed change only. Each commit shall correspond to a specific FAVOR Change Request, and a final push to the repository shall be made only once the change has been successfully made and testing performed. The commit shall contain the following information:

- Declaration that the pull request does not adversely affect the code via the listed requirements,
- Final Description of change, and
- Cases developed for testing

A documented peer review is required and shall be performed by a second independent technical reviewer (typically a software developer), if the changes are large enough that the code custodian or principal investigator cannot quickly or easily review the changes such as grammar changes or inclusion of a new unit test given as examples. If a lengthier review is required, once the peer review is completed and the agreed-upon changes are made, the code custodian or CPM (if applicable) shall approve the pull request and merge the code.

10.2 Document Control

Reviews, comment resolutions, and approvals are performed to ensure that the documents (including changes) are complete, correct and practical, satisfy the applicable requirements, and include the appropriate SQA requirements.

Distribution shall be made via SharePoint to ensure that document holders have the latest approved version.

Documents that prescribe activities affecting quality or specify SQA requirements shall be reviewed by knowledgeable reviewers, including the FAVOR SQE. Review comments shall be resolved and documented. The document shall be approved for issuance by designated approval authorities including the FAVOR SQE.

Review, comment resolutions, and approvals shall be performed to ensure that the documents (including changes) are complete, correct, and practical, satisfy applicable requirements, and include the appropriate quantitative or qualitative acceptance criteria. Changes except for editorial changes are controlled in the same way as original documents. Controls include review, comment resolution, and approval. The same organizations shall review changes as performed in the first reviews, unless otherwise designated. Reviewers have access to all information pertinent to the change.

Minor changes, such as editorial changes, do not require the same review and approval. Minor changes only require the SQE's review.

QA configuration management related records generated during the SQA process include:

- FAVOR Software Quality Assurance Plan (SQAP).
- Configuration Management and Maintenance Plan (CMMP).
- Software Requirements Document (SRD).
- Software Design Document (SDD) (if applicable).
- Software Verification and Validation Plan (SVVP).

If used, completed SQA forms including: FAVOR Code Maintenance Traveler (see

- Appendix B: FAVOR Code Maintenance Traveler Template), Software Requirements Description Criteria (see Appendix C: Software Requirements Description Criteria), Software Verification and Validation Plan Criteria (see Appendix D: Software Verification and Validation Plan Criteria), Software Design Description Criteria (see Appendix E: Software Design Description Criteria), Implementation Documentation Criteria (see Appendix F: Implementation Documentation Criteria), User Manual Criteria (see Appendix G: User Manual Criteria), Software Test Plan Criteria (see Appendix H: Software Test Plan Criteria), Software Test Results Report Criteria (see Appendix I: Software Test Results Report Criteria), and Software Verification and Validation Report Criteria (see Appendix J: Software Verification and Validation Report Criteria);
- Audits and Surveillance Reports.
- Computer Code Verification/Validation documentation that includes test plans, sample/test problems, results, verifications, validations, and reports. This documentation shall be included in the Software Test Plans (STP) and Software Test Result Reports (STRR).
- FAVLoad, FAVPFM, FAVPost source code and executables.
- FAVOR Theory Manual (which may include the SDD).
- FAVOR User's Manual.
- Relevant Training Records.

Table 10-1 summarizes the documentation requirements against the software life cycle phases, responsible authors, and interdependencies.

Table 10-1: Documentation Requirements

Process Document	Software Lifecycle Phase	Author(s)	Input Dependencies	Output Dependencies
Software Quality Assurance Plan (SQAP)	Planning	Table 3-1	SOW/NRC PM	CMMP
Configuration Management and Maintenance Plan (CMMP)	Planning	Table 3-1	SQAP	All QA related documents
Software Requirements Document (SRD)	Requirements	Table 3-1	SOW/NRC PM, SQAP	STP, SDD, SVVP
Software Verification & Validation Plan (SVVP)	Requirements	Table 3-1	SRD	STPs, SVVR
Software Verification & Validation Report (SVVR)	Testing	Table 3-1	SVVP, STRRs, JIRA	
Software Design Document (SDD) – may be a part of the FAVOR Theory Manual	Design	Table 3-1	SRD	
Software Test Plan(s) (STPs)	Testing	Table 3-1	SRD, SVVP	STRR
Software Test Results Report(s) (STRRs)	Testing	Table 3-1	STPs	SVVR
GitHub Testing Issues	Testing	Any Team Member	STPs	SVVR
Implementation Documentation 1. FAVLoad, FAVPFM, FAVPost source code and executables	Implementation/ Release	Code Developer/Code Custodian	SRD, SDD	SVVR, STP, JIRA, STRR
2. User's Manual		Table 3-1	SRD, SDD	
3. FAVOR Theory Manual		Table 3-1	SRD, SDD	
4. Acceptance Test Problems		Table 3-1	SRD, SDD	

Note: if available, the GitHub CI records may replace the STPs and STRRs, in accordance with the CMMP.

11 Verification and Validation (V&V)

This section provides a description of the test process which includes how V&V is implemented and documented in a Software Verification and Validation Plan (SVVP) (see Appendix D: Software Verification and Validation Plan Criteria). More specifically, this involves required test planning, test cases, test results, results analysis, and acceptance. Considerations include the methods and acceptance criteria used in verification through the requirements, design, implementation, and test phases of the software life cycle.

In general, the software developer and/or software tester provides a test and review plan for the model development task that includes planned testing for verification of coding, possible validation of the model with available experimental evidence, robustness testing, and performance testing (CPU). The developer also determines the level of testing to perform based on the availability of data, the available budget, and applicability of testing for the model development task.

- Review

The software developer along with the project manager determine the need for review of the model. Generally, all new models should be reviewed. Minor changes to existing models may not require review and would fall under the requirements of code maintenance.

- Verification

The developer should identify possible verification tests. This may involve unit testing (see Section 9.2) of new subroutines or integration testing (see Section 9.3) and the development of drivers for these routines. The requirements document should provide a description of the tests to be performed.

- Validation

If data exists, validation experiments should be identified. An estimate of the amount of work required in performing the validation should be made at this time. This is discussed further below.

- Robustness

The software developer and/or software tester should plan to test the new modeling over a wide range of conditions and input options to identify code failures and sensitivities. For complicated input, testing of the input should exercise every input field over a representative range. These tests should be documented as part of the final test report. If an optional new model is added to the code, the developer and/or software tester should test the code with this new model disabled to ensure that the results of the test suite are not affected.

There are many modeling options and configurations that can be used for testing. For example, the testing may be performed for cases where the Reg Guide 1.99 or EASON 2006 embrittlement correlations are used. Various ductile tearing models may also be used. Because of the wide range of available models available in FAVOR, it is impractical to test under all conditions and the developer must determine which options are important and prioritize the testing within the available budget. The developer should justify those testing conditions that are used.

As a basis for establishing a compliant V&V plan, the ASME V&V standards for computational models for solid mechanics are used in conjunction with the NUREG-BR-0167 standards. The ASME software standard guide is described in Reference [4] and the NUREG/BR-0167 software standard guide is described in Reference [2].

ASME V&V 10-2006, “Guide for Verification and Validation in Computational Solid Mechanics,” along with its illustrative associated standard ASME V&V 10.1-2012, “An Illustration of the Concepts of

Verification and Validation in Computational Solid Mechanics," provide general guidance for implementing Verification and Validation (V&V) of computational models for complex systems in solid mechanics. Since FAVOR is a software code associated with fracture mechanics, the standards are appropriate. The guidance is based on the following key principles:

1. Verification (i.e., addressing programming errors and estimating numerical errors) must precede code validation (predictive capability compared to experiments).
2. The need for validation experiments and the associated accuracy requirements for computational model predictions are based on the intended use of the model and should be established as part of V&V activities.
3. Validation of a complex system should be pursued in a hierarchical fashion from the component level to the system level.
4. Validation is specific to a computational model for an intended use.
5. Simulation results and experimental data must have an assessment of uncertainty to be meaningful.

These V&V activities establish the evidence, credibility, and confidence to ensure that FAVOR models are adequately accurate and detailed for their intended use. The standards recognize that different definitions exist for V&V within the industry. In these ASME standards, "Verification" assesses the numerical accuracy of a computational model, irrespective of the physics being modeled. Both code verification (e.g., addressing errors in the software) and calculation verification (e.g., estimating numerical errors due to under-resolved discrete representations of the mathematical model) need to be addressed. Whereas, "Validation" assesses the degree to which the computational model accurately represents the physics being modeled. Comparisons between numerical simulations and relevant experimental, analytical, or simulation data form the bases of validation. Validation activities must ascertain the predictive capability of the model in the physical realm of interest, with consideration of uncertainties that arise from both experimental and computational procedures.

Figure 11-1 and Figure 11-2 are graphical representations of the ASME definition of validation and of V&V activities and products, respectively. As shown in Figure 11-1, the V&V processes begin with a statement of the intended use of the model so that the relevant physics are included in both the model and the experiments performed to validate the model. Both modeling and experimental activities must be guided by the response features of interest and accuracy requirements for the intended use. To ensure independence, experimental outcomes for component-level, subsystem-level, or system-level tests should be provided to method modelers only after the numerical simulations have been performed with a verified model. Accounting for uncertainties in both, the V&V process ends when acceptable agreement between model predictions and experimental outcomes are achieved. If the agreement between model and experiment is not acceptable, the processes of V&V are repeated by updating the model and performing additional experiments. Currently, FAVOR has been validated against various experiments as shown in Figure 11-3 and

Figure 11-4. Funding for additional full-scale experiments are unlikely due to the high cost with little added value. These tests supported the following conclusions:

1. Cleavage fracture (both initiation and arrest) was observed in the large-scale tests consistent with small specimen data.
2. Warm pre-stressing inhibited cleavage -fracture initiation in these experiments where $K_I(t)$ is less than a previous maximum applied $K_I(t)$.
3. Observed cleavage-crack behavior in thick-section experiments were well described by Linear Elastic Fracture Mechanics (LEFM) methodology used in FAVOR.

The ASME standard emphasizes the importance of documentation in all the V&V activities. Sound documentation builds confidence and credibility in the predictive capability of computational models. Documentation also provides the historical record, traceability during audits, and captures valuable learning experiences for others.

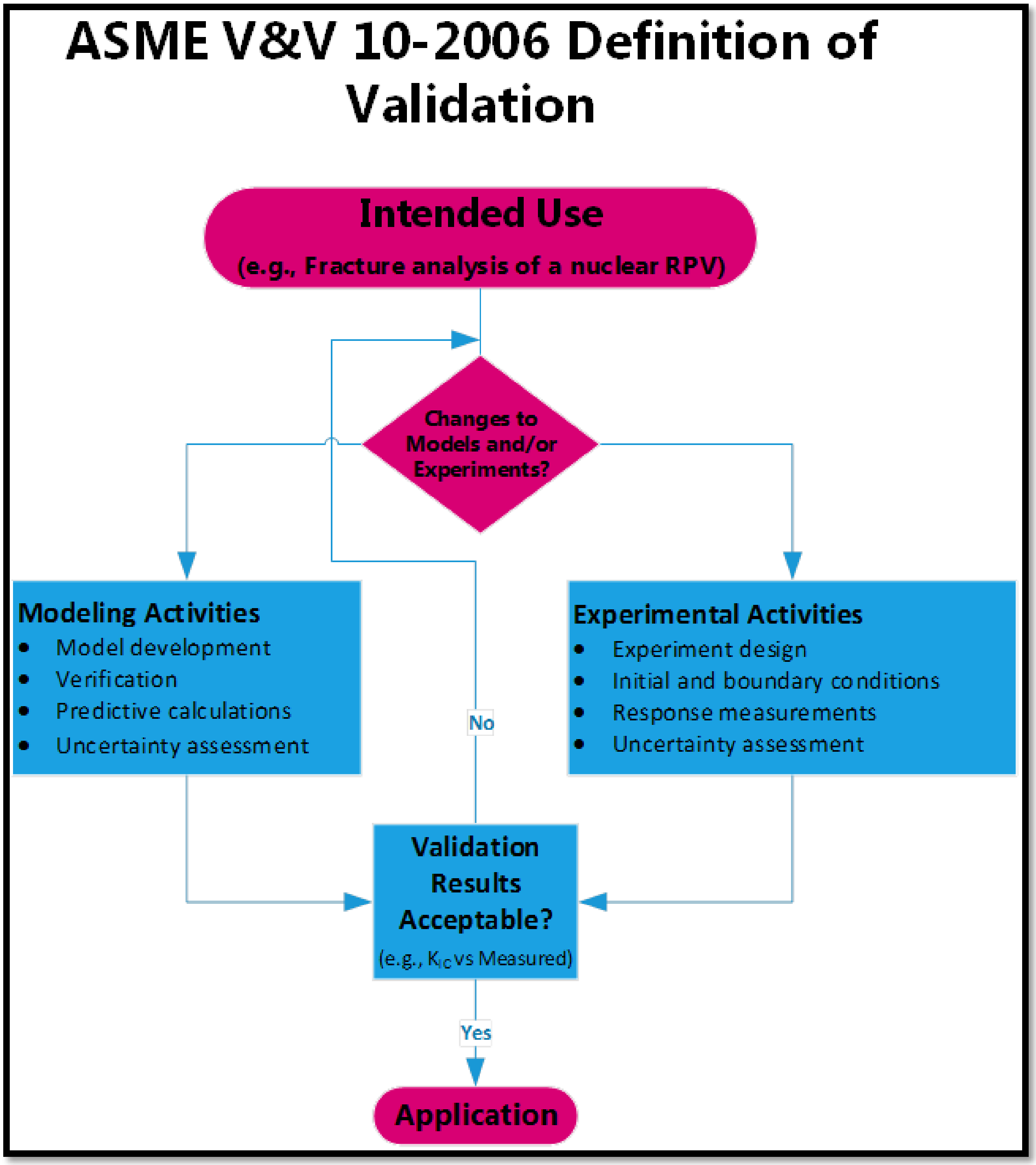


Figure 11-1: ASME V&V 10-2006 Definition of Validation

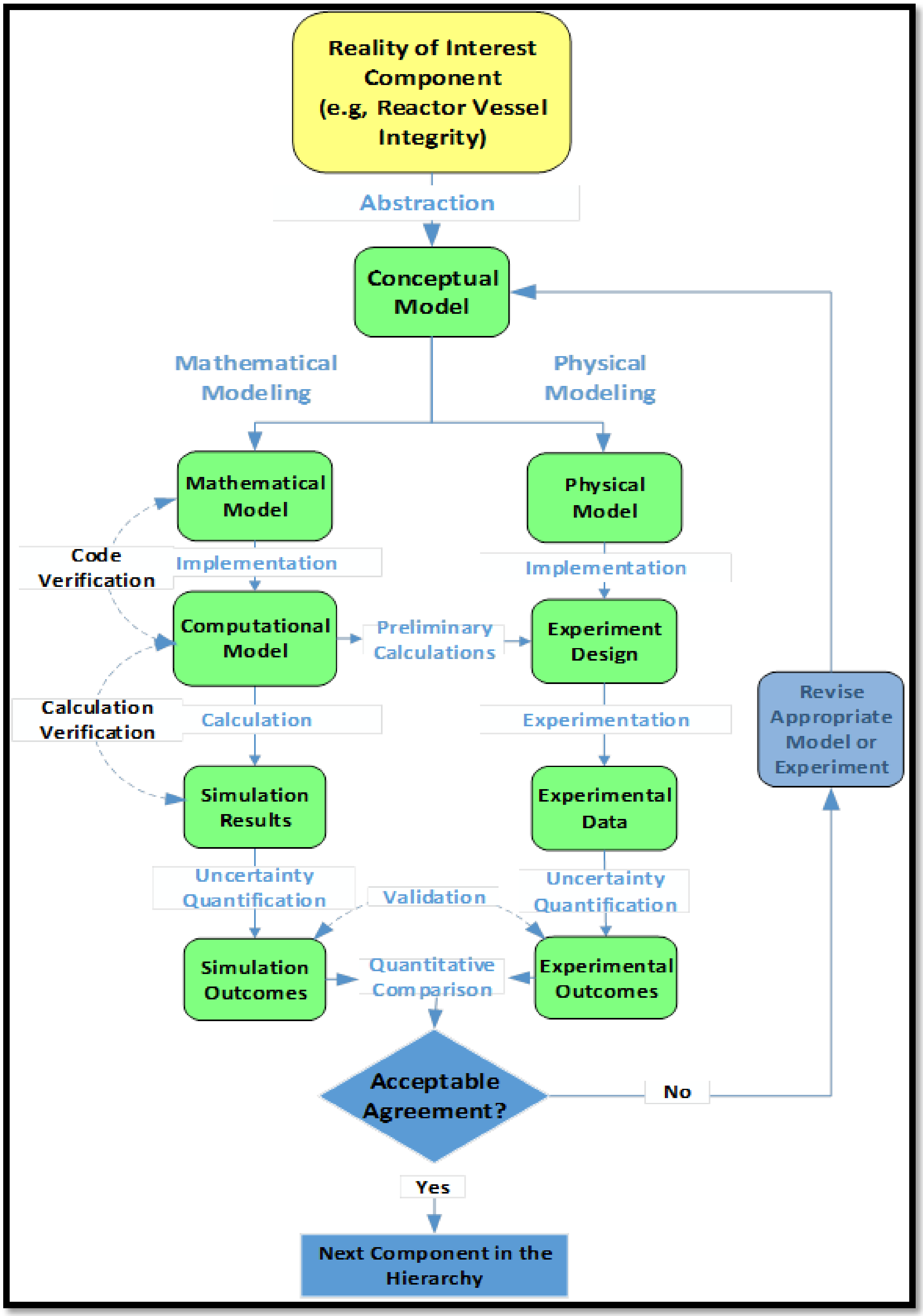
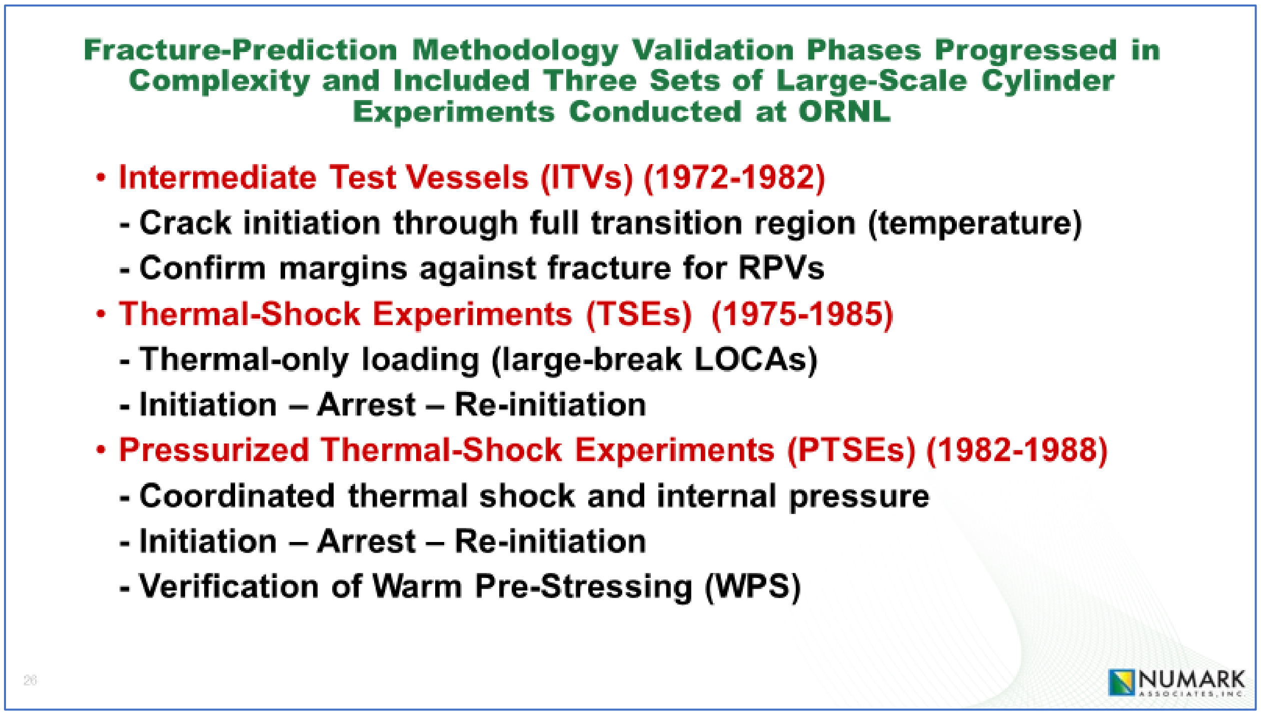


Figure 11-2: ASME V&V 10-2006 Validation and Verification Activities



Fracture-Prediction Methodology Validation Phases Progressed in Complexity and Included Three Sets of Large-Scale Cylinder Experiments Conducted at ORNL

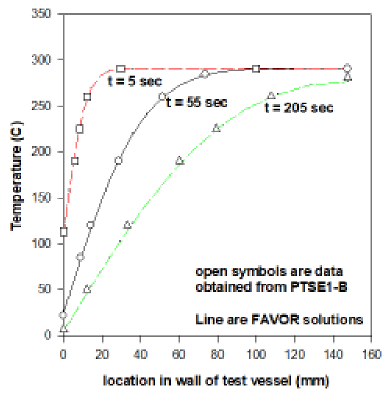
- **Intermediate Test Vessels (ITVs) (1972-1982)**
  - Crack initiation through full transition region (temperature)
  - Confirm margins against fracture for RPVs
- **Thermal-Shock Experiments (TSEs) (1975-1985)**
  - Thermal-only loading (large-break LOCAs)
  - Initiation – Arrest – Re-initiation
- **Pressurized Thermal-Shock Experiments (PTSEs) (1982-1988)**
  - Coordinated thermal shock and internal pressure
  - Initiation – Arrest – Re-initiation
  - Verification of Warm Pre-Stressing (WPS)

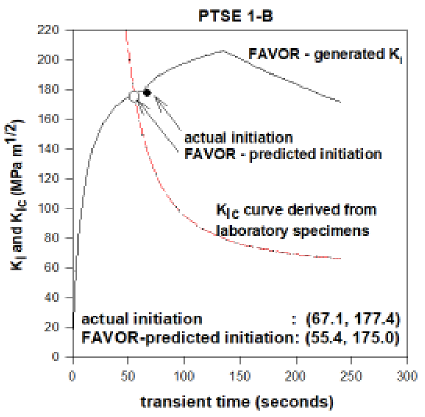
26

NUMARK ASSOCIATES, INC.

Figure 11-3: Summary of FAVOR Historical Validation Activities

FAVOR LEFM Methodology has been Validated *vs.* Large Scale Experiments





Dickson, T. L., and EricksonKirk, M. T., "Assessment of Large-Scale Pressurized Thermal Shock Experiments Using the FAVOR Fracture Mechanics Code," 2004 ASME Pressure Vessel and Piping Conference, ASME PVP volume 481, RPV Integrity and Fracture Mechanics, pp 31–37, San Diego, CA, July 2004.

31

NUMARK ASSOCIATES, INC.

Figure 11-4: FAVOR Assessment of Large-Scale PTS Experiment

As future modifications and/or maintenance to the FAVOR software occur, the software test process shall verify and validate that the software:

1. Meets the requirements that guided its design and code development,

2. Fulfills the intended use and user expectations,
3. Works as expected, and does not perform any unintended function that either by itself or in combination with other functions can degrade the entire system,
4. Can be implemented with the same characteristics, and
5. Satisfies the needs of stakeholders,
6. Relevant abnormal conditions (defects) have been evaluated for mitigating unintended functions through testing, observation, or inspection techniques. These abnormal conditions (defects) are tracked to resolution, and
7. Traceability of software requirements to software design and acceptance testing has been performed for software based on risk determination (Section 5.1).

Testing is performed and controlled to provide a high level of confidence in the validity and traceability of the resultant data. Testing to determine an item's acceptability is also controlled in order to ensure that the determinations are correct. Full details of the test procedure for a new code modification are described in the FAVOR CMMP.

Acceptance criteria are provided when testing is done for acceptance purposes, so that those performing the test are able to determine objectively whether the item is acceptable. Testing should be done by trained and qualified person(s) to ensure that the testing is done correctly, and the results are accepted as valid.

V&V tests shall be performed for every code change. Verification tests shall be designed to ensure that inputs are correctly read by the codes, and that correlations and data tables are correctly programmed into the codes. One type of verification testing is unit testing as described in Section 9.2. The second type of verification testing is running the verification test suite. Validation tests shall be designed to ensure that the FAVLoad, FAVPFM, FAVPost codes give reasonable predictions of the data in the validation test suite. Test objectives, test requirements, and acceptance criteria shall be documented and approved by the responsible design organization. Testing activities shall be controlled and have a basis described in design or other technical documents in which acceptance criteria are prescribed, as applicable.

11.1 Verification Testing

For every code change, the software tester shall develop verification test cases, to verify that the new functionality of the code performs as expected (see Appendix D: Software Verification and Validation Plan Criteria and Appendix H: Software Test Plan Criteria). In addition, the full verification test suite shall be run to verify that code functionality was not negatively impacted by the code change. The new test cases, or at least a relevant subset of the new test cases, shall be added to the verification test suite. Automation shall be used to run the full verification test suite and to check the results.

The acceptance criteria for verification testing are as follows:

- The code runs all cases to completion, and
- The predictions match expected values

The tester shall document the results in a Software Test Results Report (STRR) (see Appendix I: Software Test Results Report Criteria) and provide explanations for any differences. A second software developer shall review and approve or deny the verification testing, as appropriate. Full details of the test procedure for a new code modification are described in the FAVOR CMMP. When available, the GitHub CI records may replace the STRRs, if allowed by the CMMP.

11.2 Validation Testing

For every code change, the software tester shall develop additional validation test cases as needed, to validate the code predictions against data (see Appendix D: Software Verification and Validation Plan Criteria and Appendix H: Software Test Plan Criteria). In addition, the full validation test suite shall be run to quantify the accuracy of new code predictions. Automation shall be used to run the full validation test suite. If applicable, the new test cases shall be added to the validation test suite.

The acceptance criteria for validation testing are as follows:

- The code runs all cases to completion, and
- The code produces reasonable results relative to data.

The tester shall document the assessment in a Software Test Report (for guidance see Appendix I: Software Test Results Report Criteria), provide explanations for any differences, and the second software developer shall review and approve or deny the validation testing, as appropriate. Full details of the test procedure for a new code modification are described in the FAVOR CMMP. When available, the GitHub CI records may replace the STRRs, if allowed by the CMMP.

The PM and the PI shall concurrently be responsible for continuously monitoring the availability of new experimental data and for directing tasks to create new validation test cases that should be incorporated into the validation test suite.

11.3 Software Verification and Validation Report

The Software Verification and Validation Report (SVVR) shall document the results of implementing the SVVP. The report shall be issued at the conclusion of all V&V activities associated with the software lifecycle. The SVVR shall include the following:

- Summary of all life cycle V&V activities,
- Summary of test results including those captured in all Software Test Results Reports,
- Summary of anomalies and resolutions,
- Assessment of overall software quality,
- Lessons learned/best practices, and

- Recommendations.

The SVVR shall be developed and maintained by the Code Custodian to indicate the ability of the FAVOR software to satisfactorily perform its intended function by meeting its documented requirements.

Actual results shall be compared to established verification criteria, as documented in the SVVP and Software Test Plans, to determine acceptability. The results of the comparison shall be recorded as evidence that verification was conducted for each work product as described within the applicable lifecycle phase.

Based on the established verification criteria, work products that do not meet requirements or problems with bad verification results due to methods, procedures, criteria, or the verification environment shall be identified, and the issues recorded in JIRA to track resolution.

All the results of this analysis shall be included in the SVVR. The SVVR shall be approved by the NRC PM and the CPM (if applicable). The Software Verification and Validation Report Criteria Form (see Appendix J: Software Verification and Validation Report Criteria) can be used as an aide in developing the SVVR.

Following completion of all testing and the approval of the SVVR, the new FAVOR software version may be released.

12 Issue Reporting and Change Control

This process implements the practices and procedures to be followed for reporting, tracking, and resolving problems (corrective action) or issues identified during the software development and maintenance process and is documented in the CMMP. Errors found shall be documented and addressed using GitHub. The FAVOR GitHub repository shall be used to report and track problems or issues with the FAVOR software. The GitHub method for reporting, documenting, evaluating, tracking, and resolving of problems or issues allows for:

- A description of the evaluation process for determining whether a reported problem is an error (i.e., Bug Report) or other type of problem (e.g., Documentation Change Request, Feature Request, or Support Request).
- Definition of the responsibilities for disposition of problem reports, including notification to the originator of the results of the evaluation.
- How errors relate to appropriate configuration items.
- How errors impact past and present use of FAVOR.
- How corrective actions impact previous development activities.
- How users are notified of the identified error, impact, how to avoid the error, and pending implementation of correction actions.

When releasing new versions of FAVOR, the method shall include:

- Include a description of the change, rationale for the change, and identify the affected software baselines/versions.
- If applicable, how subsequent fixes shall be implemented.
- A FAVOR User Group Site notifies stakeholders of a new version release (only for public releases).
- States the specific organizational responsibilities concerned with their implementation.
- Changes should be formally evaluated and approved by the organization responsible for the original design unless an alternate organization has been given the authority to approve the changes.
- Only authorized changes can be made to software baselines.
- Requirements for retesting (Section 11).

During code development, this process is less formal whereby issue reporting may or may not be documented and change control is not needed to be fully implemented. Good CM practices such as versioning and baselining shall still be in place for consistency and quality integrity.

Following implementation of a new FAVOR version, any user identified errors should be reported to the NRC PM. The NRC PM, with input from the code developer and CPM (if applicable), approves the type and significance of the error along with if the FAVOR code should be changed. Software changes to address problems and corrective actions shall be documented on the FAVOR repository on GitHub in a response to the Bug Report(s). Specifically, the issue number shall be mentioned and the issue shall be closed (is possible automatically) when the corrective changes are merged into the active development branch of the code. In addition, if required, the changes shall also be reflected in the impacted CM items listed in Table 10-1 and distributed in accordance with the software release process.

Users who wish to license FAVOR under NQA-1 and 10CFR50/10CFR52 have the responsibility to review the FAVOR Software Quality Assurance and incorporate FAVOR into their own Quality Assurance Program following all applicable requirements, laws, and regulations.

13 Training

To perform acceptable work, FAVOR personnel must achieve and maintain proficiency in the technical and quality aspects of their job function. Assignment of work is based on knowledge and experience commensurate with the complexity of the task to which they are assigned. Specific qualification and training requirements for activities other than those mentioned above is determined on a case-by-case basis by the NRC PM. This determination is based on the type (scope, complexity, and nature) of work to be performed, the potential effect on quality, and the applicability of other codes or standards.

All staff whose roles and responsibilities are listed in Section 3 shall be required to read and understand this plan.

Project training activities (i.e., briefings, readings, OJT, project specific training acquired via external activities) shall be documented.

14 Procurement, Tools, and Techniques

Commercial off-the-shelf (COTS) software (including freeware, shareware, or otherwise available software) considered for acquisition to be used in the software life cycle of FAVOR are considered tools.

Until applied, these tools themselves do not have an intended function. Prior to use of a tool, the NRC PM, CPM (if applicable), and Code Developer shall review the adequacy of the acquired tool; including any associated documentation (primitive baseline) against the QA requirements identified in Table 10-1: Documentation Requirements for the intended use.

Any licenses of third-party tools that are incorporated into the project as part of the final deliverable shall be reviewed. The license agreements must be checked to assure that the resulting product is distributable to the intended users, and that no undesirable restrictions are placed on the product by the license.

When COTS software is identified as necessary for FAVOR functionality, the specific application is documented in the associated Software Test Plan (see Appendix H: Software Test Plan Criteria). The critical characteristics become evident with the development of a given module and is uniquely identified and included in the test plans, test cases, and test report for that module. The test plan that identifies the "critical characteristics" as applied within FAVOR and defines a suite of appropriate tests to verify proper functionality of those features, shall be developed and documented. This test plan and the test report documenting successful execution of the test plan shall be appropriately integrated into the FAVOR SVVP and SVVR and retained as a project record.

COTS software that does not perform a critical function nor address a critical characteristic of the FAVOR functionality is exempt from these acquisition controls. Numerical Modeling software, for example, may be acquired software, but is exempt from the software life cycle requirements.

The current set of COTS software tools and techniques being used in the various sections within the plan include:

- Excel: 3rd party tabulation software used for work planning and task tracking, input development, and output post-processing and plotting.
- Open-Source GNU Fortran ("gfortran") compiler used to compile the Fortran code.
- Intel Fortran ("Intel") compiler used to compile the Fortran code.
- Numerical Algorithms Group ("NAG") compiler used to compile the Fortran code.
- Visual Studio: 3rd party software used for managing the software development files and compilers.
- Notepad++, JEdit, VIM, Atom: 3rd party text editors used for editing Fortran source code, some of which have integration with GitHub.

- CMake/CTest: an extensible, open-source system that manages the build process and software testing in an operating system and in a compiler-independent manner.
- Fortran Package Manager (FPM): an extensible, open-source system that manages the build process and software testing in an operating system and in a compiler-independent manner.
- Git: 3rd party version control tool allowing for distributed FAVOR code development and possessing an interface with GitHub for actual storage of the version-controlled FAVOR Fortean source code.
- GitHub: 3rd party cloud repository for version-controlled FAVOR Fortran source code. Access can be given to developers to access versions of the code. In addition, GitHub can be the repository of configuration management documents (e.g., FAVOR Theory manual, FAVOR User's manual, SRD, etc.).
- SharePoint: 3rd party software used for project and software documentation, file sharing, workflow, planning, and assignment of software development tasks.
- Contractor's Shared Drives: Used for project and software documentation, if applicable.
- Ford: Open-source software used in conjunction with GitHub that automatically generates Fortran documentation from comments within the code.
- DAKOTA probabilistic modeling framework for sampling schemes external to FAVOR.

Other software may be used by the developers for code development and compiling, source control, visualization, and data analysis. It is the developer's responsibility to inform the NRC COR and CPM (if applicable) that a tool that is not listed above was used. In addition, if a new tool is extensively used to perform tasks related to the development and maintenance of FAVLoad, FAVPFM, or FAVPost, it shall be added to the list above.

15 User Documentation

Documentation of the FAVOR codes currently consists of two Oak Ridge National Lab documents and two Technical Letter Reports on FAVOR V&V efforts. Documents that describe future code releases shall be published as NRC documents. These documents are:

1. **Fracture Analysis of Vessels – Oak Ridge FAVOR, 20.1.12, Computer Code: Theory and Implementation of Algorithms, Methods, and Correlations** [1]: This document describes the equations and parameters that are included in FAVLoad, FAVPFM, FAVPost, as well as the code structure and solution scheme. Also included in the document are benchmark case comparisons against ABAQUS.

2. **Fracture Analysis of Vessels – Oak Ridge FAVOR, 20.1.12, Computer Code: User's Guide** [5]: This document describes the user input for the FAVOR codes, FAVLoad, FAVPFM, and FAVPost.

3. **Compilation of Software Quality Assurance and Verification and Validation Documentation for the Fracture Analysis of Vessels – Oak Ridge (FAVOR) Software Product** [6]**:** This document captures all public domain Software Quality Assurance (SQA)

related documentation previously created for the FAVOR code that cover the history of Validation and Verification (V&V) efforts for FAVOR (Reference x).

4. **Assessment of V&V Efforts of the Fracture Analysis of Vessels – Oak Ridge (FAVOR) Software Product – Version 16.1** [7]**:** This report provides an assessment and summary of previous Software Quality Assurance (SQA) and Verification and Validation (V&V) efforts for the FAVOR software program.

16 SQAP Reviews

Software documentation shall be reviewed as described throughout this plan.

At the discretion of the NRC PM, but at least every 5 years, the PMs, PI, and SQE should review this SQAP for possible revision. The review shall be documented by a new revision to the plan.

The SQAP shall be prepared, reviewed, approved, issued, implemented, and revised, as necessary (e.g., through requirement changes), for the project. The SQAP shall identify the applicable requirements from the NRC that apply to the work covered by the plan. Before this plan is closed-out, outstanding quality related action items shall be resolved.

17 Deliverables

The high-level deliverables of this work are released versions of the FAVLoad, FAVPFM, FAVPost codes and their associated documentation. Specific deliverables that the Contractor provides to NRC to support these high-level deliverables are documented in the SOW for the contract between the Contractor and NRC.

18 Provide Software Maintenance and Support

The maintenance and support process will be implemented and documented in accordance with the FAVOR Configuration Management & Maintenance Plan (CMMP).

19 Records

Records shall be maintained to ensure availability of documented evidence of activities performed. Both the NRC and the Contractor organizations are responsible for receiving records and shall provide protection against loss, damage, or deterioration of records. They shall also provide for the identification, storage, retrieval, and final disposition of these records.

All final NRC project records shall be controlled and handled through NRC's ADAMS system.

As of the time of this plan, the current records exist in the GitHub FAVOR repository and ADAMS. Records include the source code, FAVOR Theory Manual, and FAVOR User's Guide.

For future FAVOR development, the configuration management documents discussed in Section 10 and the deliverables discussed in Section 17 of this plan shall be documented and retained to provide a history of product quality throughout the software life cycle. All SQA records shall be collected and maintained in the software data library or archival storage for the life cycle of the product or a minimum of 5 years.

20 Retirement

Once the software is released for use by the members of the users' group, it is their responsibility to retire the software as needed.

Appendix A: Baseline Configuration Items

The table below presents the Baselined Configuration Items for FAVLoad, FAVPFM, and FAVPost 20.1.z, which serves as the starting point for maintenance activities. All proposed modifications and updates to FAVOR shall be subject to formal configuration change control as described in Section 10. The latest official version of each item is available in the GitHub FAVOR Repository.

Table A-1: Configuration Items - Documents

FAVOR – FAVLoad, FAVPFM, and FAVPost Baseline Configuration Item	**Version**	**Publication Date**
Configuration Management and Maintenance Plan (CMMP)	Rev. 0	06/21/2021
Software Requirements Document (SRD)	Not yet issued	
Software Verification & Validation Plan (SVVP)	Not yet issued	
Software Verification & Validation Report (SVVR)	Not yet issued	
Software Design Document (SDD)	Not yet issued	
Software Test Plan(s) (STPs)	Not yet issued	
Software Test Results Report(s) (STRRs)	Not yet issued	
Implementation Documentation		
FAVLoad, FAVPFM, FAVPost source code and executables	20.1.12	see below
User's Manual	20.1.12	06/24/2021
FAVOR Theory Manual	20.1.12	06/24/2021
Acceptance Test Problems	20.1.12	06/04/2021

Table A-2: Configuration Items – Software - Current

Baseline Configuration File Type/Name	Version	GitHub Unique ID
Database	Not Applicable	
Executable Code		
FAVLoad.exe	20.1.12	e866455
FAVPFM.exe	20.1.12	e866455
FAVPost.exe	20.1.12	e866455
Framework		
Microsoft Windows	Not Applicable	Not Applicable
Source Code		
GitHub.com/NRC-Research/FAVPRO	20.1.12	e866455

Appendix B: FAVOR Code Maintenance Traveler Template

Status

☐ Submitted ☐ Analyzed ☐ Approved ☐ In Progress ☐ Closed

Date	Activity
MM/DD/YYYY	

Section 1 – Request (to be completed by the Custodian from FAVOR Portal submission)

Requestor	
Name	
Email	
Date	MM/DD/YYYY
Type	
☐ Problem Report ☐ Modification Request	
Description	
Subject (Brief)	
Suggested Handling Priority	☐ Critical ☐ Elevated ☐ Standard
Detailed Description:	
List impacted software and documentation items and versions (optional):	
Attach supporting information:	
Pursue Maintenance Analysis?	
☐ Request Accepted ☐ Request Rejected	
Certification (If Rejected)	
Justification	
FAVOR Custodial Lead	
[Typed Name] Signature:	Date:

Section 2 – Maintenance Analysis (to be completed by the Custodian)	
Classification of Maintenance Needed	
☐ Correction ☐ Enhancement	
Criticality	
Recommended Handling Priority	☐ Critical ☐ Elevated ☐ Standard
Problem Verification (required only for Problem Reports)	
Test Strategy (brief description)	
Test Results:	
Attach supporting information:	
Summary of Request	
Evaluation	
Value Assessment:	
Risk Analysis:	
Maintenance Options:	
1. 2. 3.	
Option Evaluation and Recommendation:	
Detailed Maintenance Plan	
Option 1	
Impacted software and documentation items and versions	
Process	
1. 2. 3. 4.	
Notes	
Custodian Staffing Plan (skillsets, hours, and costs)	
Additional Staffing Needs (skillsets and hours) and Basis	
Option # (Repeat for additional options, if requested by FAVOR NRC PM)	
Impacted software and documentation items and versions	
Process	
1.	

2. 3. 4.
Custodian Staffing Plan (skillsets, hours, and costs)
Additional Staffing Needs (skillsets and hours) and Basis
Certification
FAVOR Maintainer
[Typed Name] Signature: Date:
FAVOR Custodial Lead
[Typed Name] Signature: Date:

Section 3 – Approval (to be completed by FAVOR MCB)

Decision	
Result	☐ Approved ☐ Rejected
Date	MM/DD/YYYY
If Rejected	
Basis	
If Approved	
Approved Handling Priority	☐ Critical ☐ Elevated ☐ Standard
Resources to Provide	
Scope	
Completion Schedule	
Certification	
FAVOR Custodial Lead	
[Typed Name] Signature: Date:	
EPRI Project Contact	
[Typed Name] Signature: Date:	

NRC Project Contact
[Typed Name] Signature: Date:

Section 4 – Outputs (to be completed by the Custodian)

Classification of Maintenance Performed
☐ Correction ☐ Enhancement
Scope
List of New or Revised Configuration Items Produced:
Actual Costs
Custodian Resources (skillsets, hours, and costs):
Other Resources (skillsets and hours):
Certification
QA Administrator
[Typed Name] Signature: Date:
FAVOR Custodial Lead
[Typed Name] Signature: Date:

Appendix C: Software Requirements Description Criteria

FAVOR Software Quality Assurance	**Software Requirements Description Criteria**	**FAVOR-SQA-3**

Document Name: ______________________________ **Document Number:** __________

Author: ______________________________ **Document Version:** __________

Technical Reviewer: ______________________________

Prior to approval of the software requirements description (SRD) document, all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Functionality: Are the functions that the software is to perform completely and uniquely identified and does the SRD define its scope?	☐ **Yes**	☐ **N/A**
Performance: Are time-related software operations issues, e.g., speed, recovery time, or response time identified, where applicable as based on the code functionality?	☐ **Yes**	☐ **N/A**
Inputs and Constraints: Are elements that are required for design or that will restrict design options identified?	☐ **Yes**	☐ **N/A**
Acceptance Criteria: Is acceptance criteria specified for each defined function?	☐ **Yes**	☐ **N/A**
Attributes (non-time-related): Are attributes such as reliability, availability, maintainability, and portability identified for each defined function?	☐ **Yes**	☐ **N/A**
Interfaces		
Are appropriate interfaces, identified, including user interactions the software will provide, for each defined function?	☐ **Yes**	☐ **N/A**
Are Framework SRD requirements in agreement with the module SRD interfacing requirements?	☐ **Yes**	☐ **N/A**
Are the sources and recipients of interface data between external sources and the software accurately described?	☐ **Yes**	☐ **N/A**
Are the internal interfaces between software functions accurately described?	☐ **Yes**	☐ **N/A**
Are descriptions of protocols for transferring and receiving data across interfaces accurate?	☐ **Yes**	☐ **N/A**
Safety and Security Features: For the Framework SRD, are the following requirements complete?		
Vulnerability protection?	☐ **Yes**	☐ **N/A**
Cyber security?	☐ **Yes**	☐ **N/A**

Verifiability: Are the requirements testable and can they be verified?	☐ **Yes**	☐ **N/A**
Consistency: Are requirements concise, unambiguous, and consistent with each other?	☐ **Yes**	☐ **N/A**
Technical Feasibility: Are the requirements technically feasible and can they result in a useable code?	☐ **Yes**	☐ **N/A**
User Manual: Are the requirements for form, content, and delivery method defined?	☐ **Yes**	☐ **N/A**
References: Are applicable references, specifications, codes, standards, regulations, procedures, or instructions that establish software requirements, test, inspection, and acceptance criteria identified?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable.
Check **N/A** for items which are not applicable.

Appendix D: Software Verification and Validation Plan Criteria

FAVOR Software Quality Assurance	Software Verification and Validation Plan Criteria	FAVOR-SQA-4

Document Name: ____________________ **Document Number:** __________

Author: ____________________ **Document Version:** __________

Technical Reviewer: ____________________

Prior to approval of the Software Verification and Validation Plan (SVVP), all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Methods: Are the methods to be used to perform the verification of each work product defined?	☐ **Yes**	☐ **N/A**
Environment: Has the environment to be used for the verification of each work product been specified?	☐ **Yes**	☐ **N/A**
Acceptance Criteria: Has criteria been developed that align with each work product, requirements, methods, and characteristics of the verification environment and provide traceability to previous work products?	☐ **Yes**	☐ **N/A**
Requirements Phase V&V Does the SVVP specify requirements for a software requirements traceability analysis, software requirements evaluation, software requirements interface analysis, and system test plan?	☐ **Yes**	☐ **N/A**
Design Phase V&V Does the SVVP specify requirements for a software design traceability analysis, software design evaluation, software design interface analysis, unit test plan, integration test plan, unit test design, and integration test design?	☐ **Yes**	☐ **N/A**
Implementation Phase V&V Does the SVVP specify requirements for a source code traceability analysis, source code evaluation, source code interface analysis, source code documentation analysis, unit test cases, integration test cases, unit test procedures, integration test procedures, and unit test execution?	☐ **Yes**	☐ **N/A**
Test Phase V&V Does the SVVP specify requirements for acceptance test procedures, integration test execution, acceptance test execution, and SVVR generation?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable.
Check **N/A** for items which are not applicable.

Appendix E: Software Design Description Criteria

FAVOR Software Quality Assurance	Software Design Description Criteria	FAVOR-SQA-5

Document Name: ______________________________ **Document Number:** __________

Author: ______________________________ **Document Version:** __________

Technical Reviewer: ______________________________

Prior to approval of the software design description (SDD) document, all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Are the following appropriately defined and documented in the SDD and appropriate for the iteration of the lifecycle?		
Software architecture	☐ **Yes**	☐ **N/A**
Numerical methods	☐ **Yes**	☐ **N/A**
Mathematical models	☐ **Yes**	☐ **N/A**
Physical models	☐ **Yes**	☐ **N/A**
Control flow	☐ **Yes**	☐ **N/A**
Control logic	☐ **Yes**	☐ **N/A**
Data model	☐ **Yes**	☐ **N/A**
Data flow	☐ **Yes**	☐ **N/A**
Process flow	☐ **Yes**	☐ **N/A**
Data structures	☐ **Yes**	☐ **N/A**
Relationships between data structures and process structures	☐ **Yes**	☐ **N/A**
Does the design consider the computer program operating environment?	☐ **Yes**	☐ **N/A**
Are measures to mitigate the consequences of problems, including internal and external abnormal conditions and events addressed?	☐ **Yes**	☐ **N/A**
Are allowable or prescribed ranges for inputs and outputs specified?	☐ **Yes**	☐ **N/A**
Is the design verifiable through testing or other means?	☐ **Yes**	☐ **N/A**
Is the design consistent with and traceable to the software's requirements?	☐ **Yes**	☐ **N/A**
Is the design technically feasible?	☐ **Yes**	☐ **N/A**

Is the design presented in enough detail to allow for implementation as computer software?	☐ **Yes**	☐ **N/A**
Are Framework SDD design elements in agreement with interfaces defined in the module SDDs?	☐ **Yes**	☐ **N/A**
Are all descriptions of data items and interfaces correct, complete, and consistent?	☐ **Yes**	☐ **N/A**
Are user interfaces understandable, will detect user errors, and will provide clear error responses?	☐ **Yes**	☐ **N/A**
If a prototype has been built for a critical interface, has it been thoroughly and independently evaluated, and does it demonstrate the critical features in the design properly? This may be facilitated through user tests using a Beta version of the software.	☐ **Yes**	☐ **N/A**
Are interfaces designed for effective configuration management?	☐ **Yes**	☐ **N/A**
Have any missing interfaces been identified?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.

Appendix F: Implementation Documentation Criteria

FAVOR Software Quality Assurance	**Implementation Documentation Criteria**	**FAVOR-SQA-6**

Software Element Name: ______________________ **Document Number:** __________

Developer: ______________________ **Document Version:** __________

Technical Reviewer: ______________________

Prior to approval of the Implementation Documentation, all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Source Code		
Is the source code provided? *Note:* If the source code is not controlled in a configuration management system then a hardcopy of the source is required. (*Check "N/A" for commercially obtained software for which source code was not provided.*)	☐ **Yes**	☐ **N/A**
Are change descriptions in the source code clear and sufficient?	☐ **Yes**	☐ **N/A**
Coding Standards Are the coding standards and conventions which were adhered to in the development of the software identified?	☐ **Yes**	☐ **N/A**
Coding Standards Implementation Does the source code adhere to the coding standards and conventions defined in the Implementation Documentation?	☐ **Yes**	☐ **N/A**
Coding Suitability Is the completed code structured and written in a reasonable and appropriate manner for the intended purpose and as reliable, maintainable code suitable for integration?	☐ **Yes**	☐ **N/A**
Executable Generation Was the executable generation process documented?	☐ **Yes**	☐ **N/A**
Is the design technically feasible?	☐ **Yes**	☐ **N/A**
Is the design presented in sufficient detail to allow for implementation as computer software?	☐ **Yes**	☐ **N/A**
Are Framework SDD design elements in agreement with interfaces defined in the module SDDs?	☐ **Yes**	☐ **N/A**
Are all descriptions of data items and interfaces correct, complete, and consistent?	☐ **Yes**	☐ **N/A**

Are user interfaces understandable, will detect user errors, and will provide clear error responses?	☐ **Yes**	☐ **N/A**
If a prototype has been built for a critical interface, has it been thoroughly and independently evaluated, and does it demonstrate the critical features in the design properly? This may be facilitated through user tests using a Beta version of the software.	☐ **Yes**	☐ **N/A**
Are interfaces designed for effective configuration management?	☐ **Yes**	☐ **N/A**
Have any missing interfaces been identified?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.

Appendix G: User Manual Criteria

FAVOR Software Quality Assurance	User Manual Criteria	FAVOR-SQA-7

Document Name: ____________________ Document Number: __________

Author: ____________________ Document Version: __________

Technical Reviewer: ____________________

Prior to approval of the User Manual, all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Does the User Manual contain?		
A statement(s) of functional capabilities (consistent with those in the software requirements description document(s)) and system limitations?	☐ **Yes**	☐ **N/A**
An explanation of the mathematical model and numerical models, where applicable as based on code functionality?	☐ **Yes**	☐ **N/A**
Physical and mathematical assumptions, where applicable as based on code functionality?	☐ **Yes**	☐ **N/A**
The capabilities and limitations inherent in the software and model?	☐ **Yes**	☐ **N/A**
The identification of input parameters, formats, and valid ranges?	☐ **Yes**	☐ **N/A**
The identification and description of output specifications and formats?	☐ **Yes**	☐ **N/A**
The identification of components of the code that were not tested?	☐ **Yes**	☐ **N/A**
Computer system vulnerability protections?	☐ **Yes**	☐ **N/A**
Specification of qualified target platforms and system requirements?	☐ **Yes**	☐ **N/A**
Messages initiated as a result of errors (i.e., improper input, failure to converge, missing data, unable to write output) and how the user can respond?	☐ **Yes**	☐ **N/A**
Problem reporting methods and methods of notification?	☐ **Yes**	☐ **N/A**
Instructions that describe the user's interactions with the software?	☐ **Yes**	☐ **N/A**
A description of any required training necessary to use the software?	☐ **Yes**	☐ **N/A**
Suggested In-Use Tests?	☐ **Yes**	☐ **N/A**
Installation instructions?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.

Appendix H: Software Test Plan Criteria

FAVOR Software Quality Assurance	Software Test Plan Criteria	FAVOR-SQA-8

Document Name: ______________________ **Document Number:** __________

Author: ______________________ **Document Version:** __________

Technical Reviewer: ______________________

Prior to approval of the software test plan (STP), all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Required Tests, Test Sequences, and Staging Does the STP identify required tests, appropriate sequence, verification methods, and the stages at which testing is required and acceptance criteria to ensure the final software satisfies the requirements of the software requirements document (SRD)? Check "Yes" if peer review is identified to fulfill the validation requirements.	☐ **Yes**	☐ **N/A**
Required Ranges of Input Parameters and Assumptions Are acceptable ranges of inputs specified to assure the software performs within its defined capabilities and limitations during testing?	☐ **Yes**	☐ **N/A**
Do the test cases demonstrate that the code adequately performs all intended functions and produces valid results for problems encompassing the range of permitted usage?	☐ **Yes**	☐ **N/A**
Does user interface testing exercise the success and failure paths for each input and provide the appropriate look of the output?	☐ **Yes**	☐ **N/A**
Test Case Criteria Are the criteria for establishing test cases defined?	☐ **Yes**	☐ **N/A**
Requirements for Testing Logic Branches and Failure Paths Does the STP specify requirements for testing logic branches to ensure proper flow?	☐ **Yes**	☐ **N/A**
Does the STP specify requirements for testing failure paths to ensure there are no abnormal terminations?	☐ **Yes**	☐ **N/A**
Requirements for Hardware Integration Are all hardware interfaces identified with necessary testing to assure functionality?	☐ **Yes**	☐ **N/A**

Requirements for COTS Software Providing Necessary Functionality Is all commercial off-the-shelf (COTS) software that provides necessary functionality to the code identified and the associated critical characteristics and their capabilities and limitations for the intended use identified with necessary testing to assure functionality?	☐ **Yes**	☐ **N/A**
Anticipated Output Values Does the STP identify expected values that the code should produce and test the boundary of anticipated values to ensure that out of range values are properly handled?	☐ **Yes**	☐ **N/A**
Acceptance Criteria Does the STP identify acceptance criteria that are traceable to requirements specified in the SRD?	☐ **Yes**	☐ **N/A**
Test Result Validation Methods Check one or more, where applicable, as based on code functionality: The test results will be compared to the following:		
• Hand Calculations	☐ **Yes**	☐ **N/A**
• Manual Inspection	☐ **Yes**	☐ **N/A**
• Calculations using comparable proven problems	☐ **Yes**	☐ **N/A**
• Empirical data and information form confirmed published data and correlation and/or technical literature (e.g., Oak Ridge HSST tests)	☐ **Yes**	☐ **N/A**
• Other validated software of similar purpose (e.g., ABAQUS)	☐ **Yes**	☐ **N/A**
• Other independent software of similar purpose (e.g., ABAQUS)	☐ **Yes**	☐ **N/A**
A documented peer review will be performed.	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.

Appendix I: Software Test Results Report Criteria

FAVOR Software Quality Assurance	**Software Test Results Report Criteria**	**FAVOR-SQA-9**

Document Name: ______________________ **Document Number:** __________

Author: ______________________**Document Version:** __________

Technical Reviewer: ______________________

Prior to approval of the software test results report (STRR), all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Test Report Contents		
Does the STRR contain a summary of test results compared to expected results from the software test plan (STP) captured in the software test log?	☐ **Yes**	☐ **N/A**
Do test results meet the acceptance criteria identified in the software requirements description and STP?	☐ **Yes**	☐ **N/A**
Are test cases and test results documented in sufficient detail so they can be repeated?	☐ **Yes**	☐ **N/A**
Does the STRR contain a summary of anomalies including status, disposition, and resolutions?	☐ **Yes**	☐ **N/A**
Is the STRR written in a manner that can be understood by an independent, technically competent individual?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.

Appendix J: Software Verification and Validation Report Criteria

FAVOR Software Quality Assurance	**Software Verification and Validation Report Criteria**	**FAVOR-SQA-10**

Document Name: ______________________ **Document Number:** __________

Author: ______________________ **Document Version:** __________

Technical Reviewer: ______________________

Prior to approval of the Software Verification and Validation Report (SVVR), all items shall be appropriately addressed so that "**Yes**" or "**N/A**" may be checked.

Does the SVVR contain a summary of the V&V activities for lifecycle phase that aligns to the Software Verification and Validation Plan?		
• Requirements Phase V&V	☐ **Yes**	☐ **N/A**
• Design Phase V&V	☐ **Yes**	☐ **N/A**
• Implementation Phase V&V	☐ **Yes**	☐ **N/A**
• Test Phase V&V	☐ **Yes**	☐ **N/A**
Does the SVVR contain a summary of task results compared to expected results captured in module STRRs?	☐ **Yes**	☐ **N/A**
Do V&V activities meet the acceptance criteria identified in the SVVP allowable or prescribed ranges for inputs and outputs specified?	☐ **Yes**	☐ **N/A**
Are V&V activities documented in sufficient detail so they can be repeated?	☐ **Yes**	☐ **N/A**
Does the SVVR contain a summary of anomalies including status, disposition, and resolutions?	☐ **Yes**	☐ **N/A**
Does the SVVR contain an assessment of overall software quality?	☐ **Yes**	☐ **N/A**
Does the SVVR contain lessons learned and best practices (to include deficiencies in the V&V process)?	☐ **Yes**	☐ **N/A**
Does the SVVR contain recommendations for software acceptance?	☐ **Yes**	☐ **N/A**
Is the SVVR written in a manner that can be understood by an independent, technically competent individual?	☐ **Yes**	☐ **N/A**

Key for check boxes above:

Check **Yes** for each item reviewed and found acceptable. Check **N/A** for items which are not applicable.