Technical Letter Report
TLR-RES/DE/REB-2021-06

FAVOR Configuration Management and Maintenance Plan (CMMP)

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

Project Summary

Project Name	Subtask 1.1: Configuration Management and Maintenance Plan (CMMP)
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

Signatures

Role	Name	Signature	Date
NRC Project Manager – Approval	Patrick Raynaud	Patrick Raynaud	06/29/2021
Lead Software Developer – Preparer/Reviewer	Terry Dickson	Terry Dickson	06/29/2021
Code Custodian – Preparer/Reviewer	Patrick Raynaud	Patrick Raynaud	06/29/2021
Software Quality Representative - Reviewer	Andrew Dyszel	Andrew Dyszel	6/29/2021
Contractor PM – Approval	Marvin Smith	Marvin Smith	6/29/2021

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
IDE	Integrated Development Environment
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-1-2015)
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 Reviewer/Tester	Person sufficiently independent with respect to the material/product they are reviewing/testing, who did not perform the work they are reviewing or testing, and who 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.
Nonconformance	A deficiency in characteristic, documentation, or procedure that renders the software quality of FAVOR to be unacceptable or indeterminate.
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)

Validation	The process of evaluating software to determine whether it satisfies specified requirements, by comparing code predictions to experimental data or independent benchmark standards. Specifically, per the IEEE Std 730™-2014 standard (Reference [2]), the process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it **at the end of each life cycle activity**, solve the right problem (e.g., correctly model physical laws and use the proper system assumptions), and satisfy intended use and user needs.
Verification	Mathematical proof of the correctness of algorithms, by confirming that code subroutines and functions produce the expected numerical output as the software goes **through each life cycle activity**. As Noted in IEEE Std 730™-2014 standard (Reference [2]), "Verified" designates the corresponding status. In design and development, verification includes examining the result of a given activity to determine conformity with the stated requirement for that activity. A system may be verified to meet the stated requirements yet be unsuitable for operation by the actual users.
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 or independent benchmark standards, to quantify the accuracy, bias, and uncertainty of code predictions.

Contents

List of Tables

1 Introduction

The purpose of this Configuration Management and Maintenance Plan (CMMP) is to describe the plan used for configuration management (CM) and maintenance process of the FAVOR code modules, FAVLoad, FAVPFM, and FAVPost. Any relevant Quality Assurance (QA) procedures take precedence over this CMMP. The ASME V&V (Reference [3]), ASME NQA-1-2015 (Reference [4], specifically Part II, Subpart 2.7), and IEEE (References [2], [5], and [6]) standards along with guidance in NUREG/BR-0167 (Reference [7]) form the basis for the requirements specified in the FAVOR CMMP. These requirements are coupled with those in the FAVOR Software Quality Assurance Plan (SQAP) (Reference [8]). In order to officially release new versions of FAVOR to the public, the responsible individuals are required to follow this plan. No other official versions will be maintained under this plan without approval from the U.S. Nuclear Regulatory Commission (NRC).

The FAVOR CMMP includes some developmental requirements that a potential user/purchaser may require if licensing FAVOR under NQA-1 and 10 CFR 50 / 10 CFR 52. A user wishing to license FAVOR in this manner has the responsibility to review the FAVOR SQAP and CMMP and incorporate FAVOR into their own Quality Assurance Program following all applicable requirements, laws, and regulations. The information contained in these two documents describe the FAVOR SQA elements and serve as an information source for users considering implementing FAVOR in their QA Program.

2 Procedure

The preparation, control, and review of this plan and its revisions are described in the FAVOR SQAP (Reference [8]). The original and subsequent revisions will be approved by both contractor PM(s) and the NRC responsible manager. Approval is indicated by the responsible person's signature in the approval blocks of this Configuration Management and Maintenance Plan (CMMP).

All the FAVOR configuration management (CM) documents and files are created and maintained using the guidance within this plan unless a quality assurance procedure overrides the process herein. The forms listed in the Appendices of the SQAP provide a template/procedure to adhere to the principles of software quality assurance and thereby when completed, provide sufficient evidence that adequate software quality assurance measures are in place. These forms, or similar GitHub features, shall be used as part of the FAVOR SQA and CM processes. Some of these forms are attached to this CMMP as these are unique to the whole software development process and release of a new FAVOR code version. Forms are located on the FAVOR SharePoint site and some copies may be located on the FAVOR GitHub repository.

3 Configuration Management and Maintenance Plan (CMMP)

This software configuration management and maintenance plan (CMMP) establishes guidance 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 3.2).

Section 8 provides various tools and techniques that are used to maintain the various configuration management documents.

3.1 FAVOR Code Modification and 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. The code developer begins by requesting approval to proceed with code changes by submitting an issue in GitHub, or by completing applicable portions of the FAVOR Code Maintenance Traveler (see Template in Appendix A), and then submitting a pull request that contains the reason for the change, any impacts on the code, and a description of the change(s).

Unless warranted, the commit made to the repository on GitHub is for the proposed change only. Each pull request shall correspond to a specific FAVOR change request on GitHub or if applicable, a FAVOR Code Maintenance Traveler. As much as practicable, a push to the repository should be made only once the change has been successfully made and testing performed (see Section 6.2). Commits shall describe the changes made to the code in a precise but succinct manner. In addition, any new tests developed to support changes to the code shall be committed to the central repository and added to the Continuous Integration test suite.

if the changes are large enough that the code custodian or principal investigator cannot quickly or easily review the changes (small changes include grammar changes or inclusion of a new unit test, for example), a documented peer review is required and is performed by a second independent technical reviewer (typically a software developer). 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.

3.1.1 Configuration Change Control Process

On GitHub, when performing software changes, the code developer shall provide commit descriptions to show evidence of the following:

- Initiation, evaluation, and disposition of a change request, The GitHub repository shall be configured such that control and approval of changes are required prior to merging into the primary development or release branches.

The GitHub repository shall also be configured to impose automatic testing of new code prior to any merges into the primary development or release branches.

GitHub provides the ability to submit software changes for comment and approval. The review on GitHub can request further changes before the software change is merged.

A software baseline shall be established at the completion of each activity of the software design process. Approved changes created subsequent to a baseline shall be added to the baseline. A baseline shall define the most recently approved software configuration. For ease of traceability, each version of the code shall be identified by a release with its unique version number. Once released, major versions of the code shall become the main branch from which to generate subsequent versions of the code (e.g. FAVPFM 20.2 branches from FAVPFM 20.1)

Released versions of the code shall remain in GitHub, or its equivalent if changed, for reproducibility and traceability of changes to other versions. In other words, the complete configuration of a released software package shall not be destroyed in the configuration management tool. Released versions of the code packages, including source code, shall also be stored in the SharePoint Library outside of GitHub for the purpose of recreating a version branch in the event that accidental deletion of branches occurs due to human error.

Configuration items to be controlled as part of a baseline shall include, as appropriate:

- Documentation (e.g., software design requirements, instructions for computer program use, test plans, and results).
- Computer program(s) (e.g., source, object, backup files, executable(s)).
- Any support software.

Changes to software shall be formally documented in GitHub through the comment resolution phase and also described in general in the SRD and SDD. The documentation shall include a description of the change and the rationale for the change and identification of affected software baselines.

Significant changes to software shall be documented in the Software Release Document and only authorized changes shall be made to software baseline. The change shall be appropriately reflected in documentation, and traceability of the change to the software design requirement shall be maintained.

For software design requirements that develop after the initial Software Requirements Document is released, that are not covered under any listed requirement, a revision of the Software Requirements Document shall be made. It is therefore recommended that the Software Requirements Document be written to include items for other necessary requirements.

Appropriate V&V and acceptance testing as described in Section 6 shall be performed for the change.

3.1.2 Code Version Identification

The FAVOR modules FAVLoad, FAVPFM, and FAVPost will be identified by name and code version number based on semantic versioning from Semantic Versioning 2.0.0 | Semantic Versioning (semver.org):

Given a baseline version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when a major software change occurs,
2. MINOR version when functionality is added in a backwards compatible manner, and
3. PATCH version when backwards compatible bug fixes are incorporated.

For example: FAVPFM-20.1.2
Where:

- FAVPFM is the source code name,
- 20.1.2 is the code version number based on MAJOR.MINOR.PATCH semantic versioning standard, where:
- 20 is the MAJOR version establishing a new Baseline and release of the code,
- 1 is a MINOR version when you add functionality in a backwards compatible manner with respect to the Baseline version, and
- 2 is a PATCH version that incorporates backwards compatible bug fixes.

Each code version that is released will print a heading in its output that identifies the code version with information specified above.

The heading in its output will print the FAVOR contact information for the code and date/time of execution.

3.1.3 Code Residence Locations

As discussed in the introduction to section 3, GitHub will be utilized for CM of the FAVLoad, FAVPFM, and FAVPost source code and executables.

3.2 Document and Record 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 is made via SharePoint library to ensure that document holders have the latest approved version, backup capability is ensured, and review/comments are retrievable from prior versions.

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

Review, comment resolutions, and approvals are 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 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 and that will be included on the FAVOR SharePoint libraries 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 from the FAVOR SQAP, including: FAVOR Code Maintenance Traveler (see Appendix A), Software Requirements Description Criteria (see [8]), Software Verification and Validation Plan Criteria (see [8]), Software Design Description Criteria (see [8]), Implementation Documentation Criteria (see Implementation Documentation Criteria), User Manual Criteria (see [8]), Software Test Plan Criteria (see [8]), Software Test Results Report Criteria (see [8]), and Software Verification and Validation Report Criteria (see [8]); (NOTE: That some of these forms may also reside on the GitHub repository for FAVOR)
- 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 Reports (STR).
- FAVOR Theory Manual (which may include the SDD).
- FAVOR User's Manual.
- Relevant Training Records.

Note that the official FAVOR Source Code and Executable(s) are located on the FAVOR GitHub repository.

Table 3-1 summarizes the documentation requirements against the software life cycle phases, responsible authors, and interdependencies. 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: Documentation Requirements

Process Document	Software Lifecycle Phase	Author(s)	Input Dependencies	Output Dependencies
Software Quality Assurance Plan (SQAP)	Planning	Table 5-1	SOW/NRC PM	CMMP
Configuration Management and Maintenance Plan (CMMP)	Planning	Table 5-1	SQAP	All QA related documents
Software Requirements Document (SRD)	Requirements	Table 5-1	SOW/NRC PM, SQAP	STP, SDD, SVVP
Software Verification & Validation Plan (SVVP)	Requirements	Table 5-1	SRD	STPs, SVVR
Software Verification & Validation Report (SVVR)	Testing	Table 5-1	SVVP, STRRs, JIRA	
Software Design Document (SDD) – may be a part of the FAVOR Theory Manual	Design	Table 5-1	SRD	
Software Test Plan(s) (STPs)	Testing	Table 5-1	SRD, SVVP	STRR
Software Test Results Report(s) (STRRs)	Testing	Table 5-1	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 5-1	SRD, SDD	
3. FAVOR Theory Manual		Table 5-1	SRD, SDD	
4. Acceptance Test Problems		Table 5-1	SRD, SDD	

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)
GitHub Testing Issues
Implementation Documentation   **1. FAVLoad, FAVPFM, FAVPost source code and executables**
2. User's Manual
3. FAVOR Theory Manual
4. Acceptance Test Problems

4 FAVOR Code Release Package

The FAVOR code package is defined in the FAVOR SQAP and is shown in Table 3-1. The following files and documents are released to users in a Software Release Document:

1. On case-by-case basis only, if deemed in the interest of the NRC: FAVLoad, FAVPFM, FAVPost source code in ASCII format so normal text editors can view the source;
2. FAVLoad, FAVPFM, FAVPost executables (*.exe files);
3. FAVOR Theory Manual;
4. FAVOR User's Manual; and
5. Acceptance Test Problems along with their solutions.

4.1 Initiation of the New FAVOR Code Version Creation Process

New code versions first start during the planning process, typically initiated but not necessarily by the SOW. Results of the planning process are translated into the Software Requirements Document to outline the requirements of the new code version. Upon completion of the code development for release, including all the completion of SQAP and CMMP requirements, the Software Release Document will include the following:

1. Table identifying FAVOR Module Executables,
2. Table identifying FAVOR User Delivery File (e.g., *.zip, *.tgz files),
3. Table describing the FAVOR User Delivery Directories and Files,
4. Table of FAVOR Zip files for NRC Delivery,
5. Table describing the NRC Delivery Directories and Files,
6. Table listing of FAVOR Software Quality Assurance documents,
7. Table of FAVOR Release Related documents associated with the released version,
8. Table or description of the computer architecture used for the build or compile of the executables,
9. Table of the build compilers used,
10. Table of acceptance test suite files (input and output),
11. Table of Build Compilers Used,
12. Table of Build Test Suite Log File, and
13. Table of Tested Operating Systems.

4.2 Testing of a New Code Version

The code custodian will assure that a test version of the new modification is created by applying the approved set of Code Revisions to the current modification. The code custodian will assure that the new version is tested on each computer system it is to be released on, using the approved test cases. The test plans and test results are documented in the applicable STPs, STRRs, and if applicable, also the SVVP and SVVR, with the appropriate reviews and approvals as required by the criteria specified in the Appendix section of the SQAP (Reference [8]) (e.g., Appendix D: Software Verification and Validation Plan). Section 6 provides information regarding different test case suites and the control of them.

4.3 Review of a New Code Version

The review of a new code version entails review of all the SQA documentation associated with the development, design, and testing of the code. Consistent with the FAVOR SQAP and this document (Table 3-1), the following reports are required to ensure a new code version release is done in a quality fashion:

1. Software Requirements Document (SRD),
2. Software Design Document (SDD),
3. Software Test Plan(s) (STPs),
4. Software Verification & Validation Plan (SVVP),
5. Software Test Results Report(s) (STRRs), and
6. Software Verification & Validation Report (SVVR).

If there are no significant changes to the algorithms and models/methods, the STP and SVVP may be combined. Similarly, the STRRs and SVVR could be combined. Justification that these can be combined is provided within the documents. Other combinations may occur depending on the source changes (e.g., combination of the STP and STRR and the combination of SVVP and SVVR).

5 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 standards,
- 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.

The responsibilities are laid out in the FAVOR Software Quality Assurance Plan (Reference [8]) and not repeated herein. Overall, 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.

A summary of the project team responsibilities are shown in Table 5-1.

Table 5-1: Functional Responsibility Matrix[1]

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[2]	QA Manager[2]
FAVOR Software QA Plan (SQAP)	I, R, A	I, A	I	I, OD	I, R	I, R	P, R[4]	I, R, A
Configuration Mgmt. and Maintenance Plan (CMMP)	I, R, A	I, A	I	I, OD	I, R		P, R[4]	I, R, A
Software Requirements Document (SRD)	I, R, A	I, R	P, I, R[4]	OD	P, I, R[4]		I, R[4]	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[3] (STPs)		A	I, R[4]		I, R	P		
Software V&V Plan (SVVP)	I, R, A	I, R, A	R[4]	OD	I, R	P	I, R[4]	R, A
Software Test Results Reports[3] (STRRs)		R, A	I, R[4]	OD	I, R	P		
V&V Tests and Results Reports (SVVR)	R, A	I, R, A	R[4]	OD	I, R	P	S	S

[1] Note that this document does not meet the full requirements of this matrix as the document was not developed under a fully qualified Software QA program.
[2] 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.
[3] Per NUREG/BR-0167, these are classified as informal. The GitHub CI records (if available) may replace the STPs and STRRs.
[4] 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[2]	QA Manager[2]
Technical Reviews (e.g., assessments/surveillances)	P, I	P					S	S
Software Changes	R, A	I, R	I, R[4]		P			
Change Documents (Appendices D – L)	R, A	I, R	P, I, R[4]	OD	P		I	S
User Input Guide, Theory Manual	I, R, A	I, R	P, I, R[4]	OD	P, I, R[4]		S	S
Maintaining Problem Reporting, Corrective Action, & Change Control	R, A	R	P	OD	I		S	S
QA Records	A	I, R	R[4]	OD			S	S

6 Control of FAVOR Test Plans and Cases

Elements of IEEE Std 829-2008 (Reference [5]) provides the standard for FAVOR's software and system test documentation. These standards are consistent and support those in both NQA-1 and NUREG/BR-0167 (Reference [7]).

6.1 Test Plan Requirements

FAVOR testing supports the FAVOR life cycle processes. For effectiveness, FAVOR test activities are conducted in parallel with the software development processes, not just at the completion of development. FAVOR Test Plans consider the elements of software, hardware, interfaces, and operators/users. For instance, FAVOR Test Plans consider:

- *Environment*: The full range of system operating environment (as applicable).
- *Operators/users*: FAVOR communicates the proper status/condition of the software-based system to the user and correctly processes all user inputs to produce the required results. For incorrect user inputs, ensure that FAVOR protects user from entering into an invalid or erroneous state. In addition, testing includes any security, interface protocols, and system assumptions are consistently applied and used across each interface. Testing can also include the validation of user documentation (e.g., error messages, help files, user guides, training material, etc.).
- *Hardware*: FAVOR correctly interacts with each hardware interface and provides a controlled system response (i.e., graceful degradation) for hardware faults.
- *FAVOR Modules*: FAVLoad, FAVPFM, and FAVPost modules interface correctly with each other in accordance with the requirements, and that errors are not propagated among software components.

Ultimately, the FAVOR Test Plans are created to verify and validate with objective evidence 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,
5. Satisfies the needs of stakeholders,
6. Ensure that relevant abnormal conditions (defects) have been evaluated for mitigating unintended functions through testing, observation, or inspection techniques.
7. Ensure that these abnormal conditions (defects) are tracked to resolution,
8. Ensure that traceability of software requirements to software design and acceptance testing has been performed for software based on risk determination.

Testing is planned, 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.

Acceptance criteria are provided in the FAVOR Test Plans. How well the tests meet the acceptance criteria is reported in a test report, such that those performing the test are able to determine objectively whether the item is acceptable. Testing is done by trained and qualified person(s) to ensure that the testing is done correctly, and the results are accepted as valid.

6.2 Test Case Types and Requirements

There are four suites of test cases that pertain to the software life cycle of FAVOR, the V&V of FAVOR, and the acceptance testing by users. These four suites can be broken down as follows:

- Unit and Integration Testing for developmental activities. This suite of cases is dependent on the modifications being designed and may include new or currently existing verification and validation cases. Unit testing shall be performed on new subroutines/functions that are added to ensure that information is properly calculated. 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 in Software Test Reports before submitting the FAVLoad, FAVPFM, and/or FAVPost changes. 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.

  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 that performance is satisfactory and that no unintentional changes to key outputs such as CPI and CPF are generated. Regression testing, similar to integration testing, is also used for selective re-testing of a FAVOR module to verify that modifications have not caused unintended effects and that the FAVOR module still complies with its specified requirements.

  As a special note, NUREG-BR-0167 [7] classifies Unit Testing and Integration Testing as informal testing because a formal test plan is not required, but Unit and Integration testing results should still be documented in the STRRs. The logs from the Continuous Integration in GitHub may replace the STPs and STRRs because they provide a review mechanism for all the unit and integration testing.

- Verification Suite is the series of test cases that have already been established to test the baseline of FAVOR. These cases verify the algorithms, models, and methods used to calculate the critical FAVOR outputs (see Table 5-1 of the FAVOR SQAP [8]). As mentioned in the Unit and Integration testing, verification and validation cases may be added if new or revised models and methods are being introduced.

- Validation Suite is the series of test cases that have been used to benchmark against measured or credible independent data (e.g., ABAQUS) that have already been established to test the baseline of FAVOR. For instance, the Appendix G cases shown in the FAVOR Theory Manual [1] are a subset of FAVOR versus ABAQUS benchmark cases. The FAVOR V&V Testing is based on standards in ASME V&V 10-2006 (Reference [3]) and IEEE Std 1012™-2016 (Reference [6]).
- The last suite of test cases are the Acceptance Test cases used to ensure that users that receive the FAVOR code release package have installed the program satisfactorily on their operating systems and reproduce the results based on the baseline configuration of FAVOR. These test cases are typically a sampling of the Verification and Validation Suite of cases.

The verification, validation, and acceptance test suites are not expected to change from one FAVOR version to the next unless significant new or revised algorithms and fracture mechanic models and methods are introduced. In order to ensure adequate control of these test suites, a copy shall be maintained on the SharePoint site.

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. 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.

FAVOR test cases (i.e., unit, integral, verification, validation) shall have the following naming convention:

General template is: **CODE**_test_**type_#** with the proper extension added to the file name: .in, .out, .dat, etc.

With the following possible values:

CODE = [**LOAD**.or.**PFM**.or.**POST**]

For FAVLoad, FAVPFM, or FAVPost, respectively

Type = [**ver**.or.**val**.or.**unit**.or.**int**]

For ‘verification’, ‘validation’, ‘unit’, or ‘integration’, respectively

= [sequential integer]

To indicate the test number

Note that in most cases, unit and integration tests do not have separate input and output files. They execute small subsets of the source code, and thus do not execute procedures for reading input or producing output files. The inputs and expected outputs are encoded within the tests themselves.

Examples:

Test name	Code	Test type	Test number
LOAD_test_ver_1	FAVLoad	verification	1
LOAD_test_val_2	FAVLoad	validation	2
PFM_test_unit_1	FAVPFM	unit	1
PFM_test_int_4	FAVPFM	integration	4
POST_test_ver_3	FAVPost	verification	3

7 Issue Reporting and Change Control

The practice and procedure to be followed for reporting, tracking, and resolving problems (corrective action) or issues identified during the software development and maintenance process is presented in this section. Errors found shall be documented and addressed using the automated problem reporting and tracking system on the FAVOR repository in GitHub. The method for reporting, documenting, evaluating, tracking, and resolving of problems or issues include:

- A description of the evaluation process (Section 7.1) 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 that correct errors identified per this section, the communication sent to users include:

- A description of the change, rationale for the change, and identify the affected software baselines/versions.
- How subsequent fixes shall be implemented.
- Methods for notifying stakeholders of a new version release.
- 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 6.1).

During code development on GitHub, 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 GitHub using a Bug Report.

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.

7.1 Evaluation Process

Corrective Action in NQA-1-2015, Part I, Requirement 16 [4] requires that conditions adverse to quality shall be identified and corrected as soon as practicable. In the case of a significant condition adverse to quality, the cause of the condition shall be determined, and corrective action taken to preclude recurrence. The identification, cause, and corrective action for significant conditions adverse to quality shall be documented and reported to appropriate levels of management. Completion of corrective actions shall be verified.

This procedure does not preclude personnel working on the FAVOR development project from reporting significant problems adverse to quality directly to the NRC. It is suggested that the severity of the problem first be assessed to avoid user input errors or misuse of the code outside of its validated and documented operating space being improperly reported as a significant problem adverse to quality.

The requirements within this procedure copy many of the requirements directly from NQA-1-2015. ASME International holds the copyright to NQA-1-2015.

7.1.1 Methods for Documenting, Evaluating, and Correction

1. Any member working on the project receiving notice of a potential problem through, but not limited to, phone, e-mail, and in-person discussions shall fill out to the best of their ability the Problem Reporting Form of Appendix B or the Bug Report on the FAVOR GitHub repository
2. The individual, upon completing the Problem Reporting Form (Appendix B), shall forward the form to the NRC PM and code custodian for evaluation.
3. Problems that appear to be significant in nature shall be reported to the NRC PM and code custodian immediately with a complete or incomplete form. Incomplete forms are acceptable if the person reporting the error needs time to gather applicable information describing the problem. This prevents delay in notifying the NRC PM or code custodian of a potentially significant issue.
4. The individual will forward all remaining information of an incomplete Problem Reporting Form for significant problems to the PM and code custodian as they are made available.
5. Any project member receiving information about minor problems such as typographical errors in outputs or manuals may wait for complete information before submitting a Problem Report Form

(9Appendix B). In lieu of a Problem Report, a documentation related problem may be reported on the FAVOR repository on GitHub using its Documentation Request form.

6. It is the responsibility of the NRC PM and code custodian to evaluate the severity of the Problem Report, assign a code developer to investigate if needed, and report back to the originator the initial findings of the investigation.
7. If the examination reveals that this is a problem with the code and not user input error, the code custodian will notify the NRC PM and the following steps are required:
   - The code custodian determines the severity of the problem and if it represents a significant problem adverse to quality.
   - The NRC PM informs the appropriate contractor PMs in the event that the report contains a significant problem adverse to quality.
   - The code custodian or software quality engineer determines how the error relates to the software lifecycle and if changes to any FAVOR QA documents are necessary.
   - The code custodian determines how the error impacts past and present use of FAVOR.

   The NRC PM (or contractor PM) and code custodian tracks progress of corrective actions in response to a significant problem adverse to quality and to implement corrective action taken to preclude recurrence.

   The code custodian determines how corrective actions impact previous development activities.

   The NRC PM (or contractor PM) verifies that the prescribed corrective actions determined during analysis of the problem have been taken.

   The NRC PM (or contractor PM) notifies the originator of the report of the corrective actions taken as a result of the evaluation.

   The NRC PM (or contractor PM) notifies the FAVOR Users Group of the error identified, its impact, and how to avoid the error, and pending implementation of corrective actions. This can be done via e-mail once complete resolution is achieved.

7.1.2 NRC Reporting of Nonconformance

1. NUREG/BR-0167 has recommended information to be included in a non-conformance report.
2. The non-conformance report sent to the NRC sponsor could be the same report as that sent to the User's Group with the removal of certain items for privacy and replacement with "member of the user group" if it is a member of the User's Group. Items that may be included are:
   a. Date and time of the detection of the nonconformance.

   b. Nonconformance identification (report number).

   c. Reporting individual and organization.

   d. Reporting individual's determination of the criticality of the nonconformance.

e. Statement of the nonconformance.

f. Organization responsible for the analysis of the nonconformance.

g. Result of the analysis of the nonconformance.

h. The project's determination of the criticality of the nonconformance.

i. Organization(s) responsible for designing, implementing, and verifying the corrective action or "fix".

j. Identification of the unit(s) of code, data, or documentation in which corrective action must be taken.

k. Summary of the test case results (or other verification activity) indicating that the corrective action was successfully implemented.

l. Identification of the date or version of the products or baseline in which the correction will be included.

m. Date on which the non-conformance is closed.

3. Upon completion of the nonconformance report the NRC PM (or Contractor PM) shall have the responsibility of delivering the report to the NRC sponsor.

8 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 3-1: 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 Section 6 for guidance). 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 FAVOR SQA 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 and integrated development environments (IDE) 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 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 Fortran 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 may also contain copies 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. In addition, the main repository for configuration management documents is on a SharePoint site.
- 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.

9 References

[1] T. L. Dickson, M. L. Smith, A. Dyszel and P. A. C. Raynaud, "TLR-NRC/DE/REB-2021-03: Fracture Analysis of Vessels – Oak Ridge FAVOR, v20.1.12, Computer Code: Theory and Implementation of Algorithms, Methods, and Correlations (ML16273A033)," U.S. Nuclear Regulatory Commission, Washington, DC, USA, June 2021.

[2] IEEE Computer Society, "IEEE Standard for Software Quality," The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 2014.

[3] 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.

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

[5] IEEE Computer Society, "IEEE Standard for Software and System Test Documentation," The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 2008.

[6] IEEE Computer Society, "IEEE Standard for System, Software, and Hardware Verification and Validation," The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 2017.

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

[8] A. Dyszel and P. A. C. Raynaud, "TLR-RES/DE/REB-2021-05: FAVOR Software Quality Assurance Plan (SQAP)," U.S. Nuclear Regulatory Commission, Washington, DC, 2021.

Appendix A 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 B FAVOR Problem Reporting Form

No. ___________

(Assigned by PM)

1. DATE and TIME ______________________
2. RELEASED VERSION ID. ______Click or tap here to enter text.______
3. ORIGINATOR ____ PHONE _____/__________________
4. ERROR REPORTED: ☐ YES ☐ NO E-MAIL ________________________
5. REPORTING INDIVIDUALS ASSESSMENT: Minor/Major/Severe/Other:____________________________

Minor Example:	Typographical Errors
Major Example:	Code Crashes
Severe Example:	Code Generate Errors that are Non Conservative

6. STATEMENT OF THE PROBLEM (including input file name used):

To be completed by PM

ACTION: ☐ APPROVED ☐ DENIED DATE RECEIVED ___________________

ASSIGNED TO _____________________________

Approval for new modification _____________________________________ Date _________________

Appendix C 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.*) Are change descriptions in the source code clear and sufficient?	☐ **Yes** ☐ **Yes**	☐ **N/A** ☐ **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.