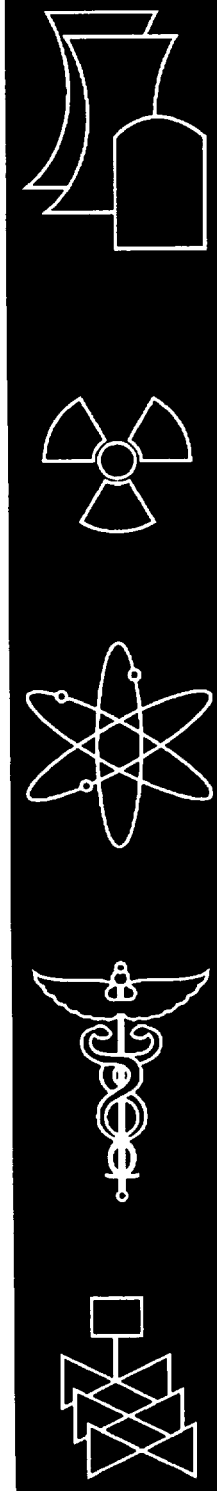
NUREG-1737



# Software Quality Assurance Procedures for NRC Thermal Hydraulic Codes

U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Washington, DC 20555-0001

## AVAILABILITY OF REFERENCE MATERIALS IN NRC PUBLICATIONS

### NRC Reference Material

As of November 1999, you may electronically access NUREG-series publications and other NRC records at NRC's Public Electronic Reading Room at www.nrc.gov/NRC/ADAMS/index.html. Publicly released records include, to name a few, NUREG-series publications; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigative reports; licensee event reports; and Commission papers and their attachments.

NRC publications in the NUREG series, NRC regulations, and *Title 10, Energy*, in the Code of *Federal Regulations* may also be purchased from one of these two sources.

1. The Superintendent of Documents
   U.S. Government Printing Office
   P. O. Box 37082
   Washington, DC 20402–9328
   www.access.gpo.gov/su_docs
   202–512–1800
2. The National Technical Information Service
   Springfield, VA 22161–0002
   www.ntis.gov
   1–800–533–6847 or, locally, 703–805–6000

A single copy of each NRC draft report for comment is available free, to the extent of supply, upon written request as follows:

Address: Office of the Chief Information Officer,
Reproduction and Distribution
Services Section
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
E-mail: DISTRIBUTION@nrc.gov
Facsimile: 301–415–2289

Some publications in the NUREG series that are posted at NRC's Web site address www.nrc.gov/NRC/NUREGS/indexnum.html are updated periodically and may differ from the last printed version. Although references to material found on a Web site bear the date the material was accessed, the material available on the date cited may subsequently be removed from the site.

### Non-NRC Reference Material

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions, *Federal Register* notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at—

The NRC Technical Library
Two White Flint North
11545 Rockville Pike
Rockville, MD 20852–2738

These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036–8002
www.ansi.org
212–642–4900

The NUREG series comprises (1) technical and administrative reports and books prepared by the staff (NUREG–XXXX) or agency contractors (NUREG/CR–XXXX), (2) proceedings of conferences (NUREG/CP–XXXX), (3) reports resulting from international agreements (NUREG/IA–XXXX), (4) brochures (NUREG/BR–XXXX), and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Directors' decisions under Section 2.206 of NRC's regulations (NUREG–0750).

NUREG-1737

# Software Quality Assurance Procedures for NRC Thermal Hydraulic Codes

Date Completed: November 2000
Date Published: December 2000

Prepared By
F. Odar

**Division of Systems Analysis and Regulatory Effectiveness**
**Office of Nuclear Regulatory Research**
**U.S. Nuclear Regulatory Commission**
**Washington, DC 20555-0001**

# ABSTRACT

This report describes quality assurance procedures for development and maintenance of the NRC thermal hydraulic codes to be used in reactor plant system transient analysis. These procedures present requirements for documentation, review, testing and assessment of the thermal hydraulic codes. These procedures will be used by the NRC staff, its contractors and partners in the code development and maintenance programs.

# TABLE OF CONTENTS

# Acknowledgments

The author is thankful for review and comments provided by the NRC staff. The NRC staff who participated in review and comment of this document were: Leo Beltracchi, Robert Brill, Terry Jackson, Vince Mousseau and Simon Smith. Special thanks are due to William Arcieri of Scientech Inc. for his discussions of Software Quality Assurance procedures.

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this document is to provide quality assurance procedures for development and maintenance of the NRC thermal hydraulic codes to be used in reactor plant system transient analysis. These procedures will be used by the NRC staff, its contractors and partners in the code development and maintenance programs.

## 1.2 Scope

Software quality assurance (SQA) is the planned and systematic actions to provide confidence that the software product meets established technical requirements. Quality assurance procedures ensure that software correctly performs all intended functions and does not perform any unintended function. SQA activities can be categorized as follows:

1) documentation of the software or software modules as they are developed,

2) verification and validation activities and their documentation,

3) nonconformance (error) reporting and corrective actions and their documentation,

4) acceptance testing and installation of the software and upgrading of code manuals,

5) configuration management, and

6) quality assessment and improvement.

This document, (Rev.0), addresses SQA activities in the first five items. The last item will be addressed in future revisions. This report is based on an internal NRC software quality assurance report with the same title designated as RPSB-99-12, written by the same author. It is a modification of the internal report. The internal report has been intended for the use of NRC staff and contractors only.

The application of SQA procedures in development of codes for NRC by DOE contractors is required by Management Directive, (M.D.), 11.7 "NRC Procedures for Placement and Monitoring of Work With the Department of Energy," Reference 1. M.D. 11.7 states, "All software development, modification, or maintenance tasks shall follow general guidance provided in NUREG/BR-0167, "Software Quality Assurance Program and Guidelines"', Reference 2. NUREG/BR-0167 provides general guidelines for development of NRC codes. This document provides procedures for development of thermal hydraulic codes. It is based on NUREG/BR-0167 and ANSI/ANS-10.4-1987, "Guidelines for the Verification and Validation of

Scientific and Engineering Computer Programs for the Nuclear Industry," Reference 3. Both standards take reference to other ANSI, ASME, IEEE and DOD standards. This document has also benefitted from other documents on quality assurance procedures prepared by Scientech, Inc. and Los Alamos National Laboratory.

For each project, an SQA plan will be developed by the project manager of the team actually developing or maintaining the code or a model. This plan will show how software quality assurance procedures will be applied to the specific project.

## 1.3 Responsibilities

### 1.3.1 Group/Program Manager - Branch Chief

The Group/Program Manager or Branch Chief is responsible for appointing a Project Manager / Principal Investigator and a Code Custodian for each code.

### 1.3.2 Project Manager / Principal Investigator

Project Manager or Principal Investigator is a person responsible for directing activities of the team where a code or model development project or a corrective maintenance activity is undertaken. Project Manager or Principal Investigator is responsible for preparation and execution of "Software Project Plan" and "Software Quality Assurance Plan," (SQAP). Guidelines on preparation of "Software Project Plan" and "Software Quality Assurance Plan" are provided in Reference 2 and in Appendix A, respectively.

### 1.3.3 Code Custodian

The Code Custodian is responsible for maintenance of the software and its configuration control.

# 2 THE SOFTWARE LIFE CYCLE AND VERIFICATION & VALIDATION

The software life cycle provides the basis for planning and implementing a software development or maintenance project. A life cycle for a model development contains following phases:

1. Initial Planning
2. Requirements Definition
3. Software Design
4. Coding
5. Software Testing
6. Installation and Acceptance

During each development phase, specific products are developed. These products are evaluated, approved and controlled. Documents are reviewed, coding is tested and approved if test results meet acceptance criteria. Reviewing and testing of a software are basically part of verification and validation activities.

**Verification** is a process of ensuring that products developed in a phase meet the requirements defined by the previous phase. After a development work in a phase is completed, we verify that the work has been performed correctly; i. e., requirements defined for that particular phase have been fulfilled. A simple example demonstrating this process is presented below.

Let us consider development of a "break flow" model in an existing code. At the "Initial Planning" phase, NRC may consider the need for such a model. If such a model is needed, NRC will establish top level requirements. The next phase is "Requirements Definition" phase which develops a physical model which would meet top level requirements established in the previous phase. The phenomena, which would occur as the flow exits through the break, are represented in terms of a mathematical model. The mathematical model is documented and requirements for the design and accuracy of the model (acceptance criteria) are specified. In the next phase, which is the "Design" phase, the program is designed to meet the design and accuracy requirements of the mathematical model. The design specifies requirements for coding. In the "Coding" phase, the design is converted into computer instructions.

In the above example, there are three verification activities to be performed at the end each phase. Products developed at each phase are 1) Software Requirements Specifications (SRS), 2) Software Design and Implementation Document (SDID) and 3) Source Coding. The first verification activity is the review of the SRS. If the review confirms that the mathematical model represents physical phenomena and that the model is expected to calculate the flow rate with desired accuracy, the model is accepted. The second verification activity is the review of the SDID. If the design meets modeling requirements, it is accepted. The third verification

activity is the review or inspection of computer instructions to ensure that coding is correctly developed as required in the design document. In short, verification can be considered to be the proof that the computer instructions correctly represent the design, that the design correctly represents the mathematical model, and that the mathematical model correctly represents the phenomena. Verification consists of a detailed examination of products developed in each phase to ensure consistency with requirements imposed by the previous phase.

The basic tools of verification are review, inspect, and audit. In the above example, these tools were sufficient to perform an adequate verification. However, in some cases, requirements may be very complex and the expected coding may be very extensive. In some cases, pilot coding to test some ideas may be necessary. In these cases, review and inspection may not be sufficient for verification. Some testing of some modules or part of coding may be necessary. This testing will be called verification testing since it tests correctness of the work performed during a development phase.

Depending on the size of the source code, testing of units of software (e.g., subroutines) and testing of collection of related units may be required. These activities are called *unit* and *integration* testing. These tests are determined as the units are developed. Test problems are generally not evident and they are not formally planned. This means that some of these tests are not included in the Testing Plan used in preparation of the SRD. Following definitions apply:

> *Unit Testing*- It is defined as testing of a unit of software such as a subroutine that can be compiled or assembled. The unit is relatively small; e.g., on the order of 100 lines. A separate driver is designed and implemented in order to test the unit in the range of its applicability.
>
> *Integration Testing*- It is defined as testing of a collection of related units that performs an identifiable functional requirement. It may be necessary to design and implement a separate driver to test the collection of units.

Unit and integration testing are considered "verification" since the results of testing are compared against the software design requirements identified in Software Design Implementation Document (SDID). They verify that the coding is correctly developed in the coding phase.

Document reviews are conducted using checklists. In this report, checklists from ANS standards are used. Coding is inspected and tested, if necessary. Audits are performed on selected items to assure that software quality assurance procedures are followed.

Verification is performed independently by different people preferably by peers. In the example presented above, first the contractor would perform verification activities before delivering documentation and the code. Next, NRC would independently conduct verification activities. In this example, verification activities to be conducted by NRC are: 1) Review and approve SRD, 2) Review and approve SDID and 3) Test and approve the coding. Results of all verification activities are documented. The extent of verification activities will be described in the Software Quality Assurance Plan.

**Validation** is a process of testing a software and evaluating results to demonstrate that the software meets its requirements as defined in Software Requirements Document (SRD). This step validates coding. It shows that not only the coding has been developed correctly but also code models, which represent the phenomena, provide accurate results as defined by the acceptance criteria. Testing is the method for software validation. Validation testing is a combination of Qualification and Acceptance testing. Two matrices, one for qualification and the other for acceptance testing are prepared. The acceptance test matrix includes all tests in the qualification test matrix plus some other tests selected by the sponsor (NRC).

In qualification testing, results obtained from the testing are compared to results from alternative methods, such as:

1. Comparison to theoretical solutions
2. Other validated computer programs
3. Experimental results (test data)
4. Standard problems with known solutions
5. Published data and correlations

The "Qualification Testing" is also called "Developmental Assessment" in thermal hydraulics. During testing, software results are obtained using different code options and input deck nodalizations. User guidelines for the software are developed during "Qualification Testing." It is expected that if appropriate user guidelines are used, test results would meet acceptance criteria. If results do not meet the acceptance criteria, modeling of the phenomena and/or user guidelines may be deficient. NRC will be informed of the results and corrective actions will be discussed.

If results in Qualification Testing meet acceptance criteria, the code will be delivered to the NRC and installed in the NRC environment. At this point, additional acceptance testing using "Acceptance Test Matrix" may be performed using the same user guidelines established in Qualification Testing. This test matrix contains the matrix of qualification testing plus some other tests chosen by NRC. Acceptance testing is performed by people different from code developers. Preferably, it is conducted by the NRC staff in its operational environment. Formal test plans for both qualification and acceptance testing are required.

# 3 ELEMENTS OF SOFTWARE QUALITY ASSURANCE

Software quality assurance (SQA) is planned and systematic actions to provide confidence that the software product meets established technical requirements. The elements of the SQA for thermal hydraulic codes are listed below:

## TABLE 1 - ELEMENTS OF SQA

| Life Cycle | Development Product | Verification & Validation Activities | Checklist |
|---|---|---|---|
| Initial Planning | SOW, Project Plan<br>SQA Plan (SQAP) | Management Review | |
| Requirements Definition | Software Requirements Specifications (SRS) | Verification of Requirements<br>Review of test plan and acceptance criteria | QA Form 03<br>QA Form 06 |
| Software Design | Software Design and Implementation Document (SDID) | Review of Design | QA Form 04 |
| Coding | Source Code<br>Verification Testing Report | Review/Inspection of Source Code<br>Verification of Program Integration<br>Verification of Test Results | QA Form 07<br>QA Form 08<br>QA Form 05 |
| Software Testing | Validation Testing Report | Validation of Program | QA Form 05<br>QA Form 09 |
| Installation and Acceptance | Installation Package<br>Upgrading Program Documentation | Verification of Installation Package<br>Verification of Program Documentation | QA Form 12<br>QA Form 10<br>QA Form 11 |

The first column shows different phases of the software development. Associated SQA

elements are shown in the second, third and fourth columns. The second column shows development products produced in different phases. The third column identifies various verification and validation activities associated with different development products. The last column identifies the checklists which can be used to perform verification and validation activities. Appendix B contains all checklists to be used in the verification and validation process.

The first phase in the life cycle is "Initial Planning." One of the products of this phase is a set of initial requirements. These requirements are top level requirements and they are set by NRC. They are contained in the Statement of Work (SOW). The management reviews these requirements. These reviews constitute verification activities. These initial requirements state "What the software will do" and not "How the software will do."

The next phase in the life cycle is the "Requirements Definition." In this phase, initial requirements are analyzed and the question of how initial requirements will be satisfied is answered. At this level requirements' specifications are established. These requirements state "What the specifications will be." They are used in the next phase where the software is designed.

As we progress from one phase to another, each phase will produce requirements for the following phase. The level of requirements becomes lower. Requirements at one level are a subset of requirements at the previous level. They represent details of requirements in the previous level. Requirements shall have following characteristics:

- **Correctness** - Ensure that requirements are correct.
- **Necessity** - Ensure that the requirement is necessary.
- **Attainable** - Ensure that the requirement is technically feasible and there is sufficient funding to perform necessary work.
- **Completeness** - Ensure that all necessary requirements are included.
- **Unambiguity** - Ensure that requirements are interpreted the same way by all readers.
- **Consistency** - Ensure that requirements do not conflict.
- **Verifiability** - Ensure that a practical method exists to verify that each requirement is satisfied. Each requirement shall be defined such that it is capable of being verified and validated by a prescribed method (e.g., review, inspection, analysis, or testing).
- **Traceability** - Ensure that software requirements trace to the initial requirements.
- **Clarity** - Ensure that readers can easily read and understand all requirements.

A discussion on writing good requirements is presented in Reference 5.

## 3.1 Initial Planning

During the initial planning phase, the technical and managerial requirements for a particular program or modification are defined. Initial requirements' definition is the responsibility of the staff in the Office of Nuclear Regulatory Research, although contractor support may be used to aid in the planning process. There are two products produced at this stage:

1. Project Plan
2. SQA Plan (SQAP)

### 3.1.1 Project Plan

A project plan describes required software activities and contractual commitments. It is a baseline management plan. The plan contains following:

a. Project Background and Objectives
b. Description of Tasks, Responsibilities and Organization
c. Scheduling and Resources
e. Implementing SQA Plan

If the work is to be conducted by a contractor, information on the first three items is generally contained in the Statement of Work (SOW) or Request for Proposal (RFP). NRC prepares these three items. Initial requirements are contained in these three items. The plan for implementation of SQAP at the contractor site is developed by the contractor. It describes how SQAP presented in this report is implemented. NRC also performs required reviews which is also a part of SQAP. If the entire work is to be performed by the NRC staff, a project plan will be prepared by the appropriate NRC staff.

A sample Project Plan is described in NUREG/BR-0167, Appendix A, Reference 2. This plan can be used as a guidance in preparing the Project Plan.

### 3.1.2 SQA Plan

At the start of each project, a Software QA Plan (SQAP) will be completed by the Project Manager or Principal Investigator. It will show the scope of the quality assurance activities to be performed in the project and will be consistent with the Project Plan. The plan may combine some of the development products of the SQA shown in Table 1. It may emphasize or de-emphasize some of the verification and validation activities. If a certain verification or validation activity or development of some documentation is to be de-emphasized, justification should be provided in the plan. The plan will basically address project needs and it will be designed specifically for the project. If the budget is not sufficient to perform all of the necessary quality assurance work, this will be stated in the plan and items which will not be covered by the quality assurance activities, will be identified. The management will be informed of the lack of coverage. Further guidance on preparation of an SQAP is provided in Appendix A.

## 3.2 Requirements Definition

### 3.2.1 Software Requirements Specifications (SRS)

This document will be based on initial overall requirements developed by the NRC. The SRS is a technical document that shall focus on technical specifications of the software. It will clearly describe analysis of each initial requirement and develop details and specifications showing how NRC requirements will be met. Following requirements will be specified.

I. **Functional Requirements:**

If the initial requirement is development of a new capability, an analysis of this requirement will be made. The theoretical basis and mathematical model consistent with the phenomena to be modeled are described. The range of parameters over which the model is applicable is specified. The relation of phenomena to code models is described. The SRS provides a detailed description of the selected model, including its range of applicability, scalableness of the model to reactor plant applications, assessment base, and accuracy. It will discuss options considered in developing these requirements and reasons for rejection of some of the options. It should include all figures, equations, and references necessary to specify the functional requirements for the design of the software.

If the initial requirement from NRC is a modification of an existing model or coding, the SRS will describe how it was done in the past and what will be done now including new functional requirements. It will discuss options considered in developing these requirements and reasons for rejection of some of the options. It should describe the new approach and how it will be implemented. It should include all figures, equations, and references necessary to specify the functional requirements for the design of the software.

II. **Performance Requirements:**

These requirements specify performance characteristics of the software or a modification. They address following items:

1) time-related issues of software operation, such as speed, etc.,
2) accuracy issues and acceptance criteria,
3) scalability

Resolution of speed, accuracy and scalableness issues require development of a test plan and acceptance criteria. In general, SRS should contain a requirement

on accuracy of code predictions relative to the phenomena to be modeled. The code should be exercised using a test plan and results should meet acceptance criteria. A sample set of acceptance criteria is presented in Appendix C. The test plan will include a list of test problems that should provide complete coverage of all of the functional requirements. It will also discuss applicability of models to reactor systems since these models are to be tested in different scales. An example of how scaling issues are addressed, is provided in NUREG/CR-5249, "Quantifying Reactor Safety Margins," Reference 4. Note that discussion of scalableness may not be applicable to some type of code work; e.g., modernization of data structures.

The test plan is also called "Qualification Test Plan." The following information should be provided in the test plan:

a. The number and types of qualification problems to be completed,

b. The rationale for their choice, why was this problem chosen, which functional requirement does it test?

c. The specific range of parameters and boundary conditions for which successful execution of the problem set will qualify the code to meet specific functional requirements,

d. Descriptions of the code input test problems,

e. A description of what code results will be compared against (analytical solution, experimental data or other code calculation)

f. Significant features not to be tested and the reasons (for example, for complex codes, absolute qualification of every combination of options over every usable range of parameters is not practical)

g. Acceptance criteria for each item to be tested. Number and types of sensitivity calculations to be performed in order to develop user guidelines.

h. Discussion of scalableness, if applicable.

The Test Plan will address following items if applicable:

a. Compliance with software requirements
b. Performance at hardware, software, user, and operator interfaces
c. Assessment of run time, user guidelines, and acceptance criteria
d. Measures of test coverage and software maintainability
e. Hardware and software used in the testing.

### III. Design Constrains

These requirements are constraints imposed on the source code that will restrict design options.

### IV. Attributes

These requirements specify operation of the software, such as portability, control, maintainability, user-friendliness, etc.

### V. External Interfaces

These are interfaces with people, hardware, and other software, including a description of the operational environment,

### VI Input and output requirements.

These specify requirements for input and output of the software.

### 3.2.2 Verification and Validation Process

SRS reviews shall be performed by the individual designated on the SQAP in the organization where SRS has been prepared. NRC staff will perform a separate review. QA Form 03 Requirements Review Checklist and QA Form 06 Software Test Plan Review Checklist may be used as a basis for these reviews. Reviews should ensure that the requirements are complete, correct, necessary, attainable, unambiguous, traceable, verifiable, consistent and technically feasible. Review should also assure that the requirements will result in a feasible and usable code. As reviews are completed, reviewers will sign an appropriate box in QAForm 03.

## 3.3 Software Design

### 3.3.1 Software Design and Implementation Document (SDID)

The design of the software is the foundation for implementing the requirements and constrains specified in the SRS. During this phase, the software design is developed. This phase is needed for all software development projects. The areas which can provide the integrity and robustness of the software design are identified below:

- **Modular Design** - Enhance the quality of software by dividing it into manageable, more understandable sets of interrelated components with clearly defined interfaces. Modular design contributes to maintainability by minimizing the ripple effect of design changes.

- **Interface Integrity** - Ensure the correctness of the interface between software components so that errors, such as invalid protocol and data, do not occur.

- **Error Handling** - Ensure that the software is robust and able to recover from an error by following a well defined strategy.

Development of the software design is performed by the software development group, either at NRC or at the contractor's office. If necessary, a pilot computer code may be developed in order to facilitate the design. The pilot code should be tested using a verification test matrix designed for this purpose. Development of a pilot code will require NRC approval. After enough knowledge is gained from testing with the pilot code, SDID is prepared. After NRC approval of the SDID, programming of the code begins.

The SDID shall describe the logical structure, information flow, data structures, the subroutine and function calling hierarchy, variable definitions, identification of inputs and outputs, and other relevant parameters. The design document shall include a tree showing the relationship among modules and a database describing each module, array, variables, and other parameters used among code modules. The level and quality of the design documentation shall allow future modifications and improvements without having to re-engineer the program. The following shall be included in the SDID, as a minimum:

a. Descriptions of major design components related to specified requirements.

b. Technical description of a program (i.e., control flow, data flow, control logic, data structures, the routine and calling hierarchy, variable definitions, identification of inputs and outputs etc.).

c. Allowable / prescribed input and output ranges.

d. Design Implementation - Model implementation or integration brings together the source code with individual models to form an operational package to

obtain desired results.

Any tools, techniques, or methodologies which must be employed during the V&V testing, shall be noted in the SDID. All this information is used to provide requirements for the coding phase. All descriptions shall be traceable to requirements specifications identified in the previous step.

### 3.3.2 Verification and Validation Process

SDID reviews shall be performed upon completion of the SDID. Reviews shall evaluate the technical adequacy of the design approach; assure internal completeness, consistency, clarity and correctness of the software design; and verify that the design is traceable to the SRS. QAForm 04, Software Design Review Checklist, may be used as a basis for this review. At the discretion of the Project Manager a Design Review Meeting may also be held. Results of verification test problems used during pilot code programming are reviewed. As reviews are completed, appropriate boxes in QAForm 04 are signed by reviewers.

## 3.4 Coding

### 3.4.1 Development of Source Code

Software coding is implementation of design requirements of the SDID. Work on Source Code will start after SRS and SDID have been prepared, reviewed and all comments are resolved. Documentation for software developed for the NRC include: 1) the program status, 2) data input and results, 3) the code version, 4) date and time of execution and, 5) output parameter units. If printouts are not usually generated, the status will clearly be noted on the computer screen.

For coding standards in Fortran 77, Reference 6 can be used. All coding in Fortran 90 language should use standards in presented in Reference 7.

If verification testing is needed, a verification test plan with defined acceptance criteria shall be developed by the code developer. Verification testing (defined in Section 2 as *unit* and *integration* testing) checks that each code module and groups of modules meet a program design requirement. Test planning can be done in parallel with the software design. The developer shall specify, as applicable, in Computer Software Testing Cover Sheet (QAForm 05):

a. The number and types of verification problems to be completed,

b. The rationale for their choice,

c. The specific portions/options of the program and range of parameters and boundary conditions for which successful execution of the problem set will verify the code related to specific design or specification requirements,

d. Significant features not to be tested and the reasons (for example, absolute verification of every combination of options over every usable range of parameters may be not practical),

e. Acceptance criteria for each item to be tested.

The Test Plan will also address, as applicable:

a. Compliance with the Software Requirement Specification

b. Performance at hardware, software, user, and operator interfaces

c. Assessment of run time, and accuracy

d. Measures of test coverage and software maintainability

e. Hardware and software used in the testing.

Note that these tests are verification tests. They are not validation tests. The purpose of verification tests at this point is to verify that the coding developed meets the requirements specified in the SDID and that the coding is done correctly. Description of the test plan, test matrix, input decks and testing results will be documented in "Verification Testing Report."

### 3.4.2 Verification and Validation Process

The Source Code Listing or update listing shall be reviewed for the following attributes. There will be sufficient explanations in comment cards in the listing which will permit review of these attributes:

a. Traceability between the source code and the corresponding design specification - analyze coding for correctness, consistency, completeness, and accuracy

b. Functionality - evaluate coding for correctness, consistency, completeness, accuracy, and testability. Also, evaluate design specifications for compliance with established standards, practices, and conventions. Assess source code quality.

c. Interfaces - evaluate coding with hardware, operator, and software interface design documentation for correctness, consistency, completeness, and accuracy. At a minimum, analyze data items at each interface.

QAForm 07, the Code Review Checklist, may be used as the basis for this review. Reviewers will sign an appropriate box in QAForm 07.

The update listing will be line by line inspected if these updates were created manually. If updates were created using Perl Scripts, or equivalent type of script, then these scripts will be documented and, inspected line by line. Inspection is detailed examination of lines of coding by an independent reviewer. The results of inspection will be reported in an inspection report. This inspection report shall contain the SQAP number, the date of updates, full listing of the updates, identifications for these updates, Perl Scripts and statements on the results of the inspection. Inspection shall address the question of whether or not updates would perform intended function. The report shall also address the question of whether or not updates would perform an unintended function.

Reviewers will review the verification test plan and results of verification tests and signify approval by signing spaces in Computer Software Testing Cover Sheet, QA Form 05 and Verification Test Report Review Checklist, QA Form 08. If results of verification testing do not meet acceptance criteria and/or the test plan is not adequate, the coding will not be approved and it will be returned to the developer for corrections or further testing.

## 3.5 Validation Testing

### 3.5.1 Performing Testing and Preparation of Testing Report

Requirements (Acceptance criteria) for validation testing have been prepared in "Requirements Definition" phase. They are documented in SRS. Execution of the Validation (Qualification) test plan may be performed by individuals who are involved with the software development group. After code development is completed, testing shall begin. As a minimum all testing described in the SRS Test Plan will be done, however additional testing may be required by NRC. The input for the test problems shall be constructed from the description in the test plan. The results shall be plotted against the data for comparison. All testing activities shall be documented and shall include information on the date of the test, code version tested, test executed, discussion of the test results, and whether the software meets the acceptance test criteria. Results from the testing report will be incorporated into the developmental assessment (DA) manual. Test reports shall contain sufficient information as described below:

1. Test Reports shall describe the testing outlined in the Test Plan in sufficient detail to allow an engineer of comparable qualifications to understand and, if necessary, reproduce the results. The report shall include nodalization diagrams, listing of the input deck, options used in constructing the deck and justification for their selection.

2. For validation (qualification) tests, code calculated results will be compared to results obtained by alternative means (exact solution, experimental data, other code calculations). A short description of these alternative methods should be included. For example, if experimental test data are used for comparison, the report will include a short description of the test facility, phenomena observed, scaling distortions which may occur in reactor transients and, availability and accuracy of measurements. It will include graphical display of selected parameters and calculated values. It will discuss the acceptance criteria and conclude whether or not the code will meet these criteria. It will discuss any user guidelines which are essential in meeting the acceptance criteria. See Appendix C for a description of sample acceptance criteria.

3. In validation (qualification) testing, when calculated code results are compared to results obtained from alternative methods such as test data, the reasons for differences should be understood and discussed in light of acceptance criteria. User guidelines will be developed, revised or confirmed based on analysis of results. It may be necessary to perform sensitivity analyses by changing parameters in input decks in order to develop user guidelines so that acceptance criteria can be met.

4. If acceptance criteria are not met although appropriate user guidelines are used, it means that limitations of the code physical models are found during the testing process. A list of these limitations will be presented and limitations will be explained in the test report. Test failures will be reviewed to ascertain adequacy of user guidelines and the soundness of the theory and design. If user guidelines are

not adequate, sensitivity studies to improve user guidelines may be needed. Calculations should be done with NRC approval. If theory or design is faulty, modifications of requirements, design and implementation may be needed. Modifications to design may be performed with NRC approval. If acceptance criteria are unrealistic, they can be changed with NRC approval.

5. Reports shall have conclusions and recommendations if necessary. All conclusions shall be supported by analyses.

### 3.5.2 Verification and Validation Activities

Review of the Validation Test Report shall be made by an independent reviewer. QAForm 09 is the Validation Test Report Review Checklist. After the review is completed, the reviewer will sign the appropriate box in the form. The reviewer may provide an additional report discussing acceptability of the product.

## 3.6 Installation and Acceptance

### 3.6.1 Installation Package

The program installation package consists of program installation procedures, files of the program, selected test cases for use in verifying installation, and expected output from these test cases.

### 3.6.2 Acceptance Testing

NRC will conduct acceptance testing in accordance with the accepted test plan. This plan will be developed by NRC. It may include following items: 1) Test cases in the installation package, 2) Selected test cases used by the contractor in Validation or Verification testing, and 3) Additional test cases as determined by NRC. See also Section 5.2.

### 3.6.3 Upgrading Program Documentation

The existing program documentation is revised and enhanced to provide a complete description of the program. Code manuals will be produced and updated concurrently with the code development process. A set of code manuals will cover following subjects:

- Theory, Models & Correlations Manual - This manual describes the theoretical basis, derivation, averaging, discretization, and solution of the conservation equations. It also describes the theoretical basis, physical models and correlations used to represent physical phenomena and includes discussions on their ranges of applicability, scaling considerations, assessment base, and accuracy.

- User's Manual - This manual shows the user how the code should be used. It provides information from code installation to post -processing including guidelines on input model preparation. It provides guidelines for selection of nodalization and code options.

- Programmer's Manual - This manual describes the code architecture from routine calling hierarchies to variable definitions. It should show control logic, the data flow, and data structure. It should provide enough detail such that an organization external to the development team can modify the code.

- Developmental Assessment Manual - This manual demonstrates the accuracy of the code predictions of phenomena observed in separate and integral effect tests. It also contains all supporting information on verification and validation tests performed while the code is being developed and assessed. It describes the code requirements, test descriptions, test results and discusses acceptability of results. It provides a qualitative statement or a quantitative measure describing how the software meets the acceptance criteria provided that the code is used with a given set of user guidelines. It provides guidance on resolution of scaling issues so that code calculations are applicable to the reactor plant system. It also demonstrates

how the models work together through integral facility tests.

Code manuals will be updated after a major modification is made. Each modification shall provide following documentation:

1. Software requirements' specification - This document will provide information to update the Theory, Models & Correlations Manual.

2. Software Design and Implementation Document - This document will provide information to update the Programmer's Manual.

3. Listing of the Source Code Modifications - This listing will provide information to update the Programmer's Manual.

4. Verification and Validation Test Reports - Theses reports will provide information to update Qualification Testing & Developmental Assessment Manual.

The information in these documents shall be reviewed and comments, if any, be resolved before their inclusion in the manuals. The review shall be performed as part of the SQA process.

### 3.6.4 Verification of the Installation Package

Verification of the installation package ensures that all elements to install the program are available and when the program is installed, it reproduces expected results. The program is installed following the procedures. Test cases which are supplied with the installation package are run. The output is checked against the output supplied with the installation package. This ensures that the program will produce the same results as the program executing in the development environment. QA Form 12 will be used for verification of the program installation package.

### 3.6.5 Verification of Upgrading of the Code Manuals

Program documentation is revised and enhanced to reflect upgrades in the code. Programmer and User Manuals are the primary sources of information about the computer program when new upgrades are to be made. They will be used by users, maintenance programmers, and V&V personnel to understand the program's objectives, characteristics and operation. Verification of these documents is performed to ensure that program documentation is completed, that the documentation conforms established standards, and that it provides a clear and correct description of the program. Checklists for verification of Programmer and User manuals are QA Form 10 and QA Form 11. The reviewer will review the manuals, provide answers to questions and sign these forms. The reviewer may also provide and document additional comments as necessary.

# 4 ERROR CORRECTIONS AND CODE MAINTENANCE

An error is a failure of the code or its documentation to meet its requirements. All errors will be reported in QA Form 13 by the users. NRC or NRC contractor evaluation and disposition of errors will be reported in QA Form 14. Most error corrections are small (e.g., less than five subroutines affected, and no subroutines added or deleted). They require small effort; e.g., less than one staff month. They may originate from users or they may relate to problems discovered during code development. Because of their size and cost, error corrections do not require the same level of documentation as code development tasks.

All errors are evaluated for their criticality and level of importance. After the evaluation, resources required for corrective actions are identified and the impact of these corrective actions are discussed. See QA Form 14 for the documentation.

Tracking of errors and reporting their correction status, the number of errors found in the code, and criticality of open problems will be kept current.

An error correction if performed by a contractor, require only a single SQA package to be delivered to NRC. This package includes QA Forms 13 and 14. QA Form 13 requires following information:

(a) A description of the symptom of the problem (i.e., code failure method, or parameter plot). Plots will be attached to QA Form 13.

(b) A description of the root cause of the bug and a demonstration that all similar bugs have been caught.

QA Form 14 requires following information:

(a) A description of how the code was changed to correct the error, including data dictionary for major code variables added, deleted, or modified, and for each modified subroutine, a statement of the deficiency being corrected, or the functionality being added or deleted.

(b) The input deck(s) for the test problem(s). One of the test problems must address the symptom from the original user problem. Plots will be attached to QA Form 14.

(c) A documented patch files that fixes the error.

(d) A statement on potential effect of the error correction on validation of the code.

The NRC personnel should duplicate plots and verify that they are complete and reasonable. If the error correction is acceptable, the NRC personnel should approve the correction for inclusion into the configuration control system. If the correction is not acceptable, then the NRC will reject the coding for correction and send it back to the code developer.

# 5 CONFIGURATION CONTROL

Software configuration control involves updating, testing, storing, distributing and final dispositioning of the software. Configuration control is primarily performed by the code custodian.

## 5.1 Configuration Control File and Software Configuration Plan

Team members or contractors who developed a model or performed a corrective maintanance, are responsible for transmitting complete project software packages which include all updates and documentation required in the SQA work. The code custodian or project manager will ensure that all QA requirements have been met.

The code custodian is responsible for establishing and maintaining the software Configuration Control file. Access to this file shall be limited. Files and supporting documentation for each code revision shall be traceable. The code custodian shall prepare a Software Configuration Plan which shows how the files are maintained and how the changes to the code files will be performed. The plan will also show how revisions to documentation are maintained. The plan will document the configuration control file.

The configuration control file shall contain following items:

1. SQAP Number (Note that this number identifies individual updates and dates), computer hardware and operating systems for which it has been tested and any special limitation on the software.

2. Listing of all documentation with identification numbers. Electronic copies of all documentation produced as SQA procedures were applied.

3. Complete software source and update files with proper identifications, including auxiliary and library files necessary to update, compile and operate the software. Suggested naming convention for these files is presented in Table 2.

4. Directory which lists and describes the contents of all files contained in the Configuration Control file. Description of the directory structure that will be used to organize files.

5. Software commands necessary to update, compile, install, test, and operate the software. Instructions on using these commands. These changes will be performed using the CVS package. The approach is to uniquely identify code changes in the source code using CVS by a unique identification label. The naming convention for the identification label should allow a reviewer to trace code changes to revisions in the supporting documentation. The naming convention for the identification label shall be defined in the Software Configuration Plan.

6. Listing of the Acceptance test problems; i.e., Qualification test problems plus special test problems for the changes in the current version. Qualification test matrix and special test problems for the current version comprise acceptance test matrix.

7. A list of complete set of manuals describing the code and its use. Electronic files of these manuals.

TABLE 2

NAMING OF CONFIGURATION CONTROL FILES

| File Name | Naming Convention |
|---|---|
| Platform Independent Source in CVS Format | .s |
| Code Library | .lb |
| Code Fixes | .fx |
| Fortran Source File in ASCII Format | .f |
| Header Files Containing Common Blocks and Other Global Definitions | .h |
| Object Files | .o |
| Executable Files | .x |
| Library Archive Files | .a |

## 5.2 Software Acceptance

The Project Manager / Principle Investigator will determine which models and changes will be included in a new version of the software. The code custodian will create the new version. Acceptance testing will be performed by the Project Manager and acceptability of the new version will be determined after resolution of comments on acceptance testing report. After the acceptance, the code version will be archived and distributed.

The new version shall contain a block of information that includes the version name, creation date, and contents (e.g., names of updates). Acceptability is determined by successfully performing test runs in the acceptance test matrix and demonstrating that results meet acceptance criteria. The acceptance test matrix contains the qualification assessment test matrix and a list of special test problems. Special test problems are two kinds: a) Test Problems for Software Configuration Testing and b) Test Problems for Performance Testing.

Software Configuration Testing is performed to check that new or modified software is compatible with prior software versions; i.e., no interference with other updates and compiles correctly on intended platforms. This is particularly important when several updates developed by different contractors are to be combined to make a new version of the code. The matrix for configuration tests contains test cases from different *unit* and *integration* testing for different updates.

Performance test matrix contains some additional testing not included in the Qualification test matrix. The testing is performed using the user guidelines developed during qualification testing. Meeting the acceptance criteria ensures that new or modified software performs correctly with the established user guidelines. Testing of the qualification test matrix will be an automatic procedure while the other tests may not be automatic since they will be developed for the specific models and updates that the new version will contain. Testing will be performed on a set of computer platforms and operating systems as needed.

The Software Acceptance Testing Report shall be prepared by the Project Manager / Principle Investigator. The report shall be reviewed and the reviewer may use Checklist QA Form 11. Upon completion of comment resolution the Project Manager / Principle Investigator will sign appropriate location in SQA Form 01.

# 6 RECORDS

All documentation produced in applying the SQA procedures described in this document shall be maintained in both electronic (CD-ROM) and hardcopy form in NRC Headquarters and appropriate contractor or partner sites for various code versions. In addition to the SQAP number, documentation shall include Job Code Number, Task number, Task Identification (Title) and a Search Code. This will permit expeditious retrieval of any documentation or a group of documents on any part of code development including related correspondence.

Code versions which are not used shall be retired and archived. The archive shall consist of the source code and all documentation produced in support of the code including all SQA documents in a retrievable format such as CD-ROM for future reference.

# 7 REFERENCES


1. "NRC Procedures for Placement and Monitoring of Work with the Department of Energy," Management Directive, (M.D.), 11.7, May 1993.

2. "Software Quality Assurance Program and Guidelines," NUREG/BR-0167, February 1993.

3. "Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry," ANSI/ANS-10.4-1987

4. "Quantifying Reactor Safety Margins," NUREG/CR-5249, December 1989.

5. "Writing good Requirements," Proceedings of the Third International Symposium of the NCOSE, Volume 2, 1993. Also, http://www.incose.org/workgrps/rwg/writing.html

6. "Safer Subsets of Fortran 77", extracted from Appendix A of Hatton, L. (1992), "Fortran, C or C++ geophysical software development," Journal of Seismic Exploration, 1, p77-92, http://www.oakcamp.demon.co.ok/Lang.F77.

7. "Fortran 90 Handbook, Complete ANSI/ISO Reference," J. Adams, W. Brainerd, J. Martin, B. Smith, J. Wagener, published by McGraw Hill, 1992, ISBN 0-07-000406-4.

# APPENDIX - A

# GUIDANCE ON PREPARATION OF AN SQA PLAN

## A-1 Software Quality Assurance Plan (SQAP)

At the start of each project, a Software QA Plan (SQAP) will be completed by the Project Manager or Principle Investigator. It will show the scope of the quality assurance activities to be performed in the project and will be consistent with the Project Plan. The plan may combine some of the development products of the SQA shown in Table 1. It may emphasize or deemphasize some of the Verification and Validation activities. If a certain verification or validation activity or a development of a certain documentation is to be deemphasized, a justification should be provided in the plan. The plan will basically address project needs and it will be designed specifically for the project. If the budget is not sufficient to perform necessary quality assurance items, this will be stated in the plan and items which will not be covered by the quality assurance activities, will be identified. If non-coverage of any item cannot be justified in the plan , the management will be informed of the non-coverage.

QAForm 01 is a SQAP Table. It shows a summary of implementation of the plan. The items in the table are discussed below.

## A-2 SQAP Number

The SQAP will be assigned a number by the Project Manager / Principle Investigator. This number shall be in form of SQAPxxxxxx.W.VRN.YYMMUP, where:

xxxxxx is the code name such as TRAC-P, TRAC-B and RELAP5.

W represents an indicator for the status of the code. There are three different statuses for the code: Production, Released, Developmental. Definitions of these statuses are given below:

Production- The code is fully qualified in accordance with this procedure and uncertainties of predictions are quantified. W replaced by P.

Released- The code is fully qualified in accordance with this procedure. All verification and validation activities are completed; all documentation is completed and the code is ready for release. The code is released for further assessment by independent parties. W is replaced by R.

Developmental- The code is partially qualified, but it is in the process of complying with this procedure. This status is used for new development of the code undergoing many updates. W is replaced by D.

VRN represents the version of the code that the work will start; such as V25. After the work is completed the version number will be updated; such as V26.

YYMMUP represents the update number(s). The format is that the first two numbers indicate the year that updates started and the next two numbers indicate

the month that updates started. The last two characters indicate identification of updates that this SQAP is to cover.

For example, if a contractor is going to improve the break flow model in the TRAC-P code and the work is going to start using version 11 in March 1998, the SQAP number would be SQAP.TRAC-P.D.V11.9803BR. In this case, BR indicates that all updates are related to the break flow model improvement.

### A-3 Review and Schedules

Reviewers' names and completion dates of reviews will be indicated in QAForm 01 (SQAP Table). If a checklist is used for the review, the number of the checklist will be entered in QAForm 01. All checklists, if used, contain reviewer's name and signature and the date of completion of the review. If there are comments that need to be resolved, QAForm 02 should be used. The date of resolution of comments is entered both in QAForm 01 and QAForm 02. All documents shall have an ID number which will permit easy retrieval.

# APPENDIX - B

# CHECKLISTS

QA Form 01 SQAP TABLE SQAP#_____

| Item | Preparer | Reviewer | Reviewer (NRC) | Checklist | Comments Resolved? | Completion Date | Document ID# |
|---|---|---|---|---|---|---|---|
| 1. Project Plan | | | | | | | |
| 2. Software Requirements Specification (SRS) inc. Qualification Assessment Test Plan | | | | | | | |
| 3. Software Design Implement. Doc. (SDID) | | | | | | | |
| 4. Verification Testing Plan and Test Report | | | | | | | |
| 5. Source Code Listing | | | | | | | |
| 6. Validation Test Report | | | | | | | |
| 7. Installation Package | | | | | | | |
| 8. Code Manuals | | | | | | | |

**QA FORM 02**
**Computer Software V&V Review Comments SQAP #_____**

| |
|---|
| Software Name: Version: |
| Documentation I.D. Reviewed: |
| Reviewer: |
| Review based on: __ checklist (attach) __ other (explain method of review) |
| Review Comments: |
| Reviewed By: _____ Date _____<br>Independent Reviewer |
| Response to Review Comments: |
| Response By: _____ Date _____ |
| Response Accepted: _____ Date _____<br>Independent Reviewer<br><br>Accepted: _____ Date _____<br>Project Manager<br><br>(required for testing only |

**QA Form 03**

**Requirements Review Checklist - SQAP #_____Doc. ID_____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Does the SRS conform to the initial requirements specified by the NRC? | | | |
| 2. Are the requirements correct for the problem to be solved? | | | |
| a. Are all requirements consistent with the Project Plan? | | | |
| b. Do requirements model the phenomena expected to occur in the transients specified by the NRC? | | | |
| c. Are the specified models, numerical techniques and algorithms appropriate for the problem to be solved? | | | |
| d. Will the program as specified solve the problem? | | | |
| e. Are descriptions of inputs and outputs correct? | | | |
| f. Do the requirements for models, algorithms and numerical techniques agree with standard references, where applicable? | | | |
| 3. Are the requirements clear and unambiguous? | | | |
| a. Can each requirement be interpreted in only one way? | | | |
| b. Are the requirements clearly organized and presented? | | | |
| c. Are program requirements clearly distinguished from other information that may be contained in the SRS? | | | |
| 4. Are the requirements necessary and complete? | | | |
| a. Do requirements include all functions called for or implied by the Project Plan? | | | |
| b. Is the operational environment (hardware, operation system) of the program specified? If applicable, are timing and sizing constraints identified? | | | |
| c. Are design constraints specified? | | | |
| d. Does the specification include desired quality requirements, such as portability, maintainability, user friendliness? | | | |
| e. If the program is required to interface with other programs, is its behavior with respect to each defined? | | | |
| f. Are input and output requirements identified and described to the extent needed to design the program? | | | |
| 5. Are the requirements internally consistent? | | | |
| a. Is the SRS free of internal contradictions? | | | |
| b. Are the specified models, algorithms and numerical techniques mathematically compatible? | | | |
| c. Are input and output formats consistent to the extent possible? | | | |
| d. Are the requirements for similar or related functions consistent? | | | |
| e. Are input data, computations, output, etc. required accuracies compatible? | | | |
| f. Are the requirements consistent with the properties of the specified operating environment, and any other programs with which the program must interface? | | | |

| | | | |
|---|---|---|---|
| 6. Are the requirements feasible? | | | |
| a. Are the specified models, algorithms and numerical techniques practical? Can they be implemented within system and development effort constraints? | | | |
| b. Are the required functions attainable within the available resources? | | | |
| c. Can the desired attributes be achieved individually and as a group? (ie, generally not possible to maximize both efficiency and maintainability.) | | | |
| 7. Do the requirements make adequate provision for program V&V testing? | | | |
| a. Is each requirement testable? | | | |
| b. Are acceptance criteria specified? | | | |
| c. Are the acceptance criteria consistent with at least one of the following (circle all appropriate): Results obtained from similar computer programs; Solutions of classical problems; Accepted experimental results; Analytical results published in technical literature; Solutions of benchmark problems | | | |

**QA Form 04**

**Software Design Review Checklist - SQAP#_____Doc. ID_____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Does the SDID conform to the requirements specified in SRS? | | | |
| 2. Is the SDID traceable to the SRS? | | | |
| a. Are all requirements implemented in the design? | | | |
| b. Are all design features consistent with the requirements? | | | |
| c. Do design features provide modularity? | | | |
| e. Are the specified numerical techniques appropriate for the problem to be solved? | | | |
| f. Are the specified algorithms appropriate for the problem to be solved? | | | |
| g. Is the structure of the design appropriate for the problem to be solved? | | | |
| h. Will the program as designed meet the requirements? | | | |
| 3. Is the design clear and unambiguous? | | | |
| a. Can all design information be interpreted in only one way? | | | |
| b. Is the design information clearly organized and presented? | | | |
| c. Is the design sufficiently detailed to prevent misinterpretation? | | | |
| 4. Is the design complete? | | | |
| a. Are all program inputs, outputs, and database elements identified and described tothe extent needed to code the program? | | | |
| b. Does the program design conform to its required operational environment? | | | |
| c. Are all required processing steps included? | | | |
| d. Are all possible outcomes of each decision point designed? | | | |
| e. Does the design account for all expected situations and conditions? | | | |
| f. Does the design specify appropriate behavior in the face of unexpected or improper inputs and other anomalous conditions? | | | |
| g. Are coding standards specified or referenced as applicable? | | | |
| h. Is the design sufficiently robust to be able to recover from an error by following a well defined strategy? | | | |
| i. If interface is required with other programs, is this provided for in the design? If applicable, does the design provide for reading and writing of external files? | | | |
| 5. Is the design internally consistent? | | | |
| a. Is the SDID free of internal contradictions? | | | |
| b. Are the specified models, algorithms and numerical techniques mathematically compatible? | | | |
| c. Are input and output formats consistent to the extent possible? | | | |
| d. Are the designs for similar or related functions consistent? | | | |
| e. Are the accuracy and units of inputs, database elements and outputs that are used together in computations or logical decisions compatible? | | | |
| f. Are the style of presentation and level of detail consistent throughout the SDID? | | | |
| 6. Is the design correct? | | | |

| | | | |
|---|---|---|---|
| a. Is the design logic sound, such that the program will do what is intended? | | | |
| b. Does the design logic provide interface integrity? | | | |
| c. Is the design consistent with the properties of the specified operation environment, and with any other programs with which the program must interface? | | | |
| d. Does the design correctly accommodate all required inputs, outputs and database elements? | | | |
| e. Do the models, algorithms and numerical techniques used in the design agree with standard references, where applicable? | | | |
| 7. Is the design attainable and technically feasible? | | | |
| a. Are the specified models, algorithms and numerical techniques practical? Can they be implemented within system and development effort constraints? | | | |
| b. Can functions, as designed, be implemented within the available resources? | | | |
| 8. Do design features permit testing (verifiability) of the requirements? | | | |

**QA Form 05**

**Computer Software Testing Cover Sheet - SQAP#_____Doc. ID_____**

| | |
|---|---|
| I.<br>Software Name: | Version: |
| Code Author and Affiliation:<br>Computer Type: | Program Language: |
| Applicable Testing: _ Verification Testing<br>_ Validation Testing | Code Verifier:<br>Code Validator: |
| II. Testing Plan Scope: | |
| III. Approved: | Date: |
| IV. Summary and Conclusion of Testing Activity: | |

**QA Form 06**
**Software Test Plan Review Checklist - SQAP#_____Doc. ID_____**

| **Reviewer:**<br>**Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Does the SRS contain information needed as a bases for testing? | | | |
| a. Are the requirements testable? | | | |
| b. Do tests simulate the phenomena to be modeled? | | | |
| c. Are the acceptance criteria specified? | | | |
| d. Are the acceptance criteria consistent with at least one of the following (circle all that apply): Results obtained from similar computer programs; Solutions of classical problems; Accepted experimental results; Analytical results published in technical literature; Solutions of benchmark problems | | | |
| 2. Do documented test plans include reference to documents containing the requirements to be tested? | | | |
| 3. Are requirements to be tested identified, with acceptance criteria ? | | | |
| 4. Are the planned test cases adequate? | | | |
| a. Is the basis for selection of test cases documented? Is the rationale clear and valid? | | | |
| b. Does each test case have known and accepted results? | | | |
| c. Are dependencies between test cases identified? | | | |
| d. Is the application range of the software product, as defined by the requirements, adequately covered by the set of test problems? | | | |
| e. If test cases are experimental results, are there sufficient number of measurements with an acceptable accuracy to perform code assessment? | | | |
| 5. Is each testable requirement adequately covered? | | | |
| a. Is at least one test case provided for each requirement? | | | |
| b. If the requirement covers a range of values or capabilities, are test cases identified to cover the range adequately? | | | |
| c. Does documentation include demonstration of test to requirements, as in a traceability matrix? | | | |
| d. Do the tests include cases that are representative of the conditions under which the program will be used? | | | |
| e. Does the test plan address applicable scaling issues? | | | |
| 6. Are the test case specifications complete? | | | |
| a. Are the test cases consistent with the planned cases that are listed? | | | |
| b. Is the specification for each test case complete? | | | |
| unique identification providing traceability of the requirement | | | |
| function(s) tested/objective(s) | | | |
| input, including modeling assumptions | | | |

| | | | |
|---|---|---|---|
| expected results | | | |
| test setup instructions | | | |
| hardware and software environment | | | |
| 7. Is the specification for each test case adequate? | | | |
| a. Is input detail sufficient? | | | |
| b. Are expected results explicit, and specified with sufficient accuracy? | | | |
| c. Do evaluation criteria provide clear acceptance criteria for each test? | | | |
| d. Are all relevant databases, data files or libraries identified? | | | |
| 8. Does the test planning provide for test databases or data files? | | | |
| 9. Are test cases specified in sufficient detail for future reproducibility? | | | |
| 10. Are instructions provided for disposition of test files and test results? | | | |
| 11. Is configuration management provided for test databases, data files, and external programs? | | | |
| 12. Can the planned testing be performed within the available resources? | | | |

**QA Form 07**

**Code Review Checklist - SQAP#_____Doc. ID_____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Does the Source Code conform to applicable standards? | | | |
| 2. Are sufficient comments provided to give an adequate description of each routine? | | | |
| 3. Is the Source Code clearly understandable? | | | |
| a. Is ambiguous or unnecessarily complex coding avoided? | | | |
| b. Is the code formatted to enhance readability? | | | |
| 4. Are all features of the design, including modularity fully and correctly implemented in the code? | | | |
| 5. Can all features of the coded program be traced to requirements in SRS and SDID? | | | |
| 6. Are all variables properly specified and used? | | | |
| a. Is the program free of unused variables? | | | |
| b. Are all variables initialized? | | | |
| c. Are array subscripts consistent? | | | |
| d. Are loop variables within bounds? | | | |
| e. Are constants correctly specified? | | | |
| f. Are proper units used with each variable? | | | |
| 7. Is there satisfactory error checking? | | | |
| a. Are input data checked for applicable range? | | | |
| b. Are external data files checked to assure that the correct data file is being read and the data are in proper format? | | | |
| c. Are results of calculations checked for reasonable values? | | | |
| d. Are error messages clear and unambiguous? | | | |
| 8. Do all subroutine calls transfer data variables correctly? | | | |
| 9. Is there sufficient evidence to verify that the processing of data and transmission of data between modules is correct? | | | |
| 10. Is the code status (PRODUCTION, VERIFIED, RELEASE or DEVELOPMENTAL indicated on the program output? | | | |

**QA Form 08**

**Verification Test Report Review Checklist -SQAP#_____Doc. ID_____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Do unit test results show that: | | | |
| a. Each major logical path within the routine was tested? | | | |
| b. Each routine was checked for appropriate minimum, maximum, and average sets of variables? | | | |
| c. Do results meet acceptance criteria? | | | |
| 2. Do integral test results meet acceptance criteria? | | | |
| 3. Does the code conform with the resource requirements on the operating system? | | | |
| a. Does the code meet storage requirements for memory and external devices? | | | |
| b. Does the code meet timing and sizing requirements? | | | |
| 4. Does the code interface properly with external files? | | | |
| 5. Are all elements of the code properly identified? | | | |
| a. Has the source code been verified? | | | |
| b. Has the compiler been identified? | | | |
| c. Have special user libraries been verified? | | | |
| d. Have system libraries been identified? | | | |
| 6. Does the program link correctly? | | | |
| 7. Are the interfaces between functional units correct? | | | |
| 8. Is the control language used for execution proper? | | | |
| 9. Are the data libraries that are used in the code appropriate? | | | |

**QA Form 09**

**Validation Test Report Review Checklist - SQAP#_____Doc. ID_____**

| **Reviewer:** **Date:** | Yes | No | N/A |
|---|---|---|---|
| 1. Does the Software V&V Report meet the requirements of SRS? | | | |
| a. Do test results corresponding to each requirement adequately cover the range? | | | |
| b. Has each test result for its associated requirement satisfied its acceptance criteria? | | | |
| c. Does the combination of test case results for the specified requirement meet the acceptance criteria? | | | |
| d. Are there any test results that indicate unrepeatable, unreliable or unexpected program behavior? | | | |
| 2. If test cases were supplied with the installation package, did they produce results identical to the output supplied with that package? | | | |
| a. Could all the test cases be performed? | | | |
| b. Were all results identical to previous results? | | | |
| c. Were differences in results clearly understood and justified? | | | |
| 3. Does testing comply with the test planning documentation? | | | |
| a. Is a summary of test results provided? | | | |
| b. Is program performance with respect to requirements evaluated? | | | |
| c. Are recommendations provided for acceptance of the code or development of further user guidelines and/or model development, as appropriate? | | | |
| d. Are results of each test case reported in detail? | | | |
| e. Were unexpected occurrences documented, as well as expected pass/fail results based on acceptance criteria? | | | |
| f. Are problems and their resolution documented? | | | |
| g. Are test planning documents, and any other relevant documents, referenced? | | | |
| h. Are deviations from test plans, if any, described and justified? | | | |
| i. Is the test environment (location, hardware configuration, support software) completely and accurately described? | | | |
| a. Is the program tested completely identified? | | | |
| 4. Does the documentation of the test results accurately reflect the testing performed? | | | |
| 5. Are all the test cases executed correctly? | | | |

| | | | |
|---|---|---|---|
| a. Do test results adequately identify the software and hardware under test, including support software such as operation system, test drivers, test data? | | | |
| b. Do reported test results indicate performance of each test case in the specified environment, using the documented specifications? | | | |
| c. Is there an explanation for any deviation from the specified test environment or procedures? | | | |
| d. Is there a problem report for each deviation from expected results? | | | |
| e. Were correct input data used for each test case? | | | |
| f. Is the output of each test case accurately reported or attached? | | | |

**QA Form 10**

**User's Manual Review Checklist - SQAP# _____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Is the level of detail in the manual appropriate for its intended users? | | | |
| 2. Does the manual describe the program's functions, options, limitations and accuracy? | | | |
| 3. Is the description of user input adequate? | | | |
| a. Are all input data requirements specified? | | | |
| b. Are formats fully specified? | | | |
| c. Are valid ranges of input values specified? | | | |
| d. Are theoretical limitations specified? | | | |
| e. Are units specified? | | | |
| 4. Are the necessary run instructions provided? | | | |
| 5. Does the manual provide guidance on preparation of input decks? | | | |
| 6. Does the manual provide guidance for interpreting output? | | | |
| 7. Are error messages adequately explained? | | | |
| 8. Are hardware and operation system requirements specified? | | | |

**QA Form 11**
**Programmer's Manual Review Checklist - SQAP# _____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Is the information in the programmer's manual consistent with other manuals? | | | |
| a. Is the user information consistent with the programmer's information? | | | |
| b. Is the information in the theory manual an accurate reflection of the coded program? | | | |
| c. Is the information in the manual clear, unambiguous and well organized? | | | |
| d. Is the manual free of internal contradictions? | | | |
| 2. Are all the elements of the program (e.g., source, library, compiler) properly identified? | | | |
| 3. Are instructions (including control language) provided for compiling, loading and running the program? | | | |
| 4. Is the design correct? | | | |
| a. Is the design logic sound, such that the program will do what is intended? | | | |
| b. Is the design consistent with the properties of the specified operating environment, and with any other programs with which the program must interface? | | | |
| c. Does the design correctly accommodate all required inputs, outputs and database elements? | | | |
| d. Do the models, algorithms and numerical techniques used in the design agree with standard references, where applicable? | | | |

**Q/A Form 12**
**Verification of the Installation Package SQAP#_____**

| **Reviewer:** **Date:** | Yes | No | NA |
|---|---|---|---|
| 1. Are sufficient materials available on the program installation tape to permit rebuilding and testing of the installed program? | | | |
| a. Are the necessary elements from the following list available? | | | |
| Source Code | | | |
| User-supplied library routines | | | |
| Module linkage specifications | | | |
| External file structure definitions | | | |
| Control Language for Installation | | | |
| External Data Libraries to be used by the program | | | |
| Test Cases | | | |
| Control Language for Execution | | | |
| Output Produced by the Test Cases. | | | |
| b. Are the format and content of the tape properly identified in the installation procedures for easy reading of the files? | | | |
| c. Are the installation procedures clearly understandable to allow installation and checkout? | | | |
| 2. Can the program be rebuilt from the installation package? | | | |
| a. Can the program source be recompiled and reloaded in the same manner as before? | | | |
| b. If there are changes in rebuilding, do these changes affect the functional operation of the program? | | | |
| 3. Do the test cases produce results identical to output supplied with the installation package? | | | |
| a. Can all test cases be performed? | | | |
| b. Are all results identical to previous results? | | | |
| c. Are differences in results clearly understood and justified? (such as new date and time on printed output) | | | |

**QA Form 13** **Trouble Report -Reporting**

**Trouble Report No.**_____
(to be entered by NRC)

To report a problem, error, or code deficiency, enter all of the following information.

Code Name:

Version:

Date:

Submitted by (name):

Submitted by (organization):

Address:

Phone Number:

E-mail Address:

Classification of the Problem or Deficiency: (Check one or more)

___ Input Processing Failure
___ Code Execution Failure
___ Restart/Renodalization Failure
___ Unphysical Result
___ Installation Problem
___ Other

Provide following items:

1. Input deck(s)
2. Description of the symptom of the bug
3. Plots, if available

Computer Hardware Type / Computer Operating System (include version):

User's Determination of the Criticality of the Problem:

**QA Form 14** **Trouble Report - Disposition**

**Trouble Report No.**_____

NRC's or Contractor's Determination of the Criticality of the Problem:

Organization Assigned for Corrective Action:

Provide following items:

1. A description of the root cause of the bug and a demonstration that all similar bugs have been caught.

2. A description of how the code was changed to correct the bug, including data dictionary for major code variables added, deleted, or modified, and for each modified subroutine, a statement of the deficiency being corrected or the funtionality being added or deleted. Following format should be used:

Subroutine Report

Subroutines Deleted:

subroutine1.f: statement of subroutine's function and why that functionality is no longer necessary, or what subroutines now perform the function. Refer to other subroutines with their full name (e.g. TempM.f).

subroutine2.f: similar statement.

Subroutines Added:

subroutine5.f: statement of subroutine's function.

subroutine8.f: statement of subroutine's function.

Subroutines Modified:

subroutine23.f: statement of what changes are being made, the deficiency being corrected, or the functionality being added or deleted.

3. Input deck(s) for test problem(s). One of the test problems must address the symptom from the original user problem.

4. A documented patch file that fixes the bug.

Date on which Nonconformance is Closed:

# APPENDIX - C

## A SAMPLE SET OF ACCEPTANCE CRITERIA

## C.1 Acceptance Criteria

Acceptance criteria or success metrics are used to judge how well the code is validated. This appendix presents an example of development of acceptance criteria where the criteria are developed for comparison of code predictions with experimental data. There are qualitative and quantitative metrics. First, acceptance criteria using quantitative metrics will be addressed.

The recently completed RELAP5 adequacy assessment effort presents important concepts used defining these criteria. Part of these concepts are repeated here for convenience. Additional concepts on providing "User Guidelines" are added here.

1) "Excellent agreement" applies when the code exhibits no deficiencies in modeling a given behavior. Major and minor phenomena and trends are correctly predicted. The calculated results are judged to agree closely with the data. The calculations will, with few exceptions, lie within the specified or inferred. uncertainty bands of the data. The code may be used with confidence in similar applications. The term "major phenomena" refers to phenomena that influence key parameters, such as rod cladding temperature, pressure, differential pressure, mass flow rate, and mass distribution. Predicting the major trends means that the prediction shows the significant features of the data. Significant features include the magnitude of a given parameter through the transient, slopes, and inflection points that mark significant changes in the parameter.

2) "Reasonable agreement" applies when the code exhibits minor deficiencies. Overall, the code provides an acceptable prediction. All major trends and phenomena are predicted correctly. Differences between calculated values and data are greater than are deemed necessary for excellent agreement. The calculation will frequently lie outside but near the specified or inferred uncertainty bands of the data. However, the correct conclusions about trends and phenomena would be reached if the code were used in similar applications. The code models and/or facility model noding should be reviewed to see if improvements can be made.

3) "Minimal agreement" applies when the code exhibits significant deficiencies. Overall, the code provides a prediction that is not acceptable. Some major trends or phenomena are not predicted correctly, and some calculated values lie considerably outside the specified or inferred uncertainty bands of the data. Incorrect conclusions about trends and phenomena may be reached if the code were used in similar applications. If the agreement is "Minimal", model noding must be reviewed and sensitivity studies using different noding or options should be performed. The selection of nodalizations or options should not be arbitrary. It should be aimed to model the phenomena more accurately. If agreement can be improved and can be reclassified as "Reasonable" or "Excellent", new "User Guidelines" should be developed. If agreement cannot be improved, it should be reclassified as "Insufficient agreement" described below.

4) "Insufficient agreement" applies when the code exhibits major deficiencies. The code

provides an unacceptable prediction of the test data because major trends are not predicted correctly. Most calculated values lie outside the specified or inferred uncertainty bands of the data. Incorrect conclusions about trends and phenomena are almost certain if the code is used in similar applications. An appropriate warning must be issued to users. Selected code models should be reviewed and modified and if necessary, new models should be developed and assessed before the code can be used with confidence in similar applications.

Quantitative metrics define requirements for accuracy of code predictions of the test data. The metrics depend on 1) the goodness / deficiency of code models simulating the phenomena and 2) the experimental data spread. Following is an example of how the first and second element of the metrics are determined in a work performed for assessment of reflood model in the TRAC-M code using Flecht-Seaset, Run# 31504, Reference C.1.

An assessment of the TRAC-P code, an earlier version of the TRAC-M code, concluded that modeling of the grid spacers were inadequate; and therefore, the option to include grid spacer models should not be used, Reference C.2. Tests performed with and without grids show that grid spacers contribute substantially to cooling of the cladding of the rods, Reference C.3. Grid spacers interact with the fuel rods and the two-phase flow in the flow channels. From various two-phase flow studies, it has been observed that grid spacers caused enhancement of the heat transfer, mainly by desuperheating the vapor. The effect could be classified under following three mechanisms:

1. Grid rewet
2. Convective enhancement
3. Droplet breakup

Since the grids are unpowered, they can quench before the fuel rods. If the grids rewet, they create additional liquid surface area that can help desuperheat the vapor in the nonequilibrium two-phase droplet flow. A wetted grid will have a higher interfacial heat transfer coefficient in comparison to droplets because the relative velocity for the vapor flow relative to the liquid film is larger. In addition to desuperheating the vapor, the liquid film will evaporate, resulting in higher steam flow and convective heat transfer. The increased interfacial heat transfer between the grid and the vapor flow and the generation of additional saturated vapor from the liquid film on the grid will result in lower vapor temperatures downstream of the grids. In addition, the grids can also break up the entrained droplets into smaller ones thereby increasing the surface area for evaporation. The evaporation of the smaller droplets will provide an additional steam source which increases the convective heat transfer coefficient.

As discussed above, grid spacers affect the reflood phenomena in a substantial way. Any calculation without a grid spacer effect should indicate much hotter cladding temperatures for the rods. TRAC-M calculations will be biased because of absence of grid spacer effects. The best way to quantify this bias is to compare cladding temperatures with and without grid spacers in test data. References C.3 and C.4 show that this bias is approximately 100K. Hence, we should expect, at the minimum, 100K of bias in code predictions. This number can

perhaps be refined if more comparisons are made. This is an example of goodness / deficiency of code modeling affecting quantitative metrics.

The next important element in determining quantitative metrics is the experimental data spread. Experimental data spread is different from the instrument uncertainty which addresses the accuracy of measurements of an instrument. Experimental data spread or uncertainty bands of the test data occur because of the random nature of the phenomena. In the Flecht-Seaset reflood experiments, the reflood is considered to be one dimensional since the geometry of the test section and the rod bundle is one dimensional. The input deck is built based on one dimensional assumptions. The code will calculate one cladding temperature at a specific location of the rod at each time step of the calculation. The rod modeled in the input deck represents all similar rods. However, in reality, the two-phase flow in the test section has some randomness in that all water slugs are not of the same length or shape; all droplets are not of the same size or shape and their concentration vary from location to location within the test section. Hence, one should neither expect that maximum cladding temperatures occur in all rods at the same time and in the same horizontal plane although the power shape for each rod is the same, nor expect that all rods quench at the same time in the same horizontal plane. Hence, there will be a normal spread of experimental data due to random nature of the two-phase flow. Since the code will predict only one value, for an "Excellent agreement" the prediction can be at the edge of the spread. Hence, this spread will form part of the quantitative metric.

Figs. C.1 through C.4 show clad temperatures measured at different elevations and at different rods. The lowest spread occurs at lower levels. The spread increases with increasing elevation. At higher elevations droplets move randomly in many directions and their sizes and shapes vary. Clad temperatures at a horizontal plane can vary as much as 100K. These figures show the data spread or uncertainty bands of cladding thermocouple traces at certain elevations in Flecht-Seaset, Run 31504. Thermocouples are located at different rods in a horizontal plane. These give multiple measurements at an elevation. These measurements not only show a spread of 100K can occur in clad temperatures but also, they also indicate that quenching time at high elevations may vary up to 120s. Thermocouples affected by the housing wall or those giving clearly erroneous readings are not included in these plots.

In summary, a quantitative metric for clad temperature predictions is a bias of 100K and a spread of 100K. This would make a total of 200K difference in predictions. The quenching time spread is 120s.

This is an example of acceptance criteria using quantitative metrics. However, in this test similar multiple measurements are not available for other parameters such as pressure drops. For pressure drop measurements, there is only one pair of pressure taps available to measure each pressure drop between two elevations. Hence, for pressure drops it is not possible to determine the data spread or uncertainty bands caused by randomness of two-phase flow. Similarly, steam probe measurements are made at few unpowered rods at

selected elevations. In this particular example, Flecht-Seaset, Run# 31504, it is assumed that these measured quantities give average quantities for a horizontal plane. Since the spread of data cannot be determined, the assessment of these parameters can only be made qualitatively.

In qualitative assessment the conclusions are based on user's judgement and experience. The user may use similar concepts as "Excellent", "Reasonable" and "Insufficient" agreements as presented above; however, the user should carefully define the terms and justify bases for his selections of the type of agreement. The overall conclusions on acceptability of predictions are more reliable if some of the parameters can be judged on a quantitative basis.

However, similar multiple measurements are not available for other parameters such as pressure drops. For pressure drop measurements, there is only one pair of pressure taps available to measure each pressure drop between two elevations. Hence, for pressure drops it is not possible to determine the data spread or uncertainty bands caused by randomness of two-phase flow. Similarly, steam probe measurements are made at few unpowered rods at selected elevations. It is assumed that these measured quantities give the average quantities for a horizontal plane. Hence, assessment of these parameters can only be made qualitatively.

### C.2 References

C.1 "Assessment of the TRAC-M Codes Using Flecht Seaset Reflood and Steam Cooling Data", F. Odar, SMSAB-00-05, July 2000.

C.2 " TRAC-M: Fortran 77, Version 5.5, Developmental Assessment Manual, Volume I: Nonproprietary Assessment Sections", B. E. Boyack, J. F. Lime, D. A. Pimentel, J. W. Spore, J. L. Steiner, LA-UR-99-6480, December 1999.

C.3 "FEBA Flooding Experiments with Blocked Arrays, Data Report 1, Test Series I through IV, KfK 3658, March 1984.

C.4 "FEBA Flooding Experiments with Blocked Arrays, Data Report 2, Test Series V through VIII, KfK 3649, March 1984.

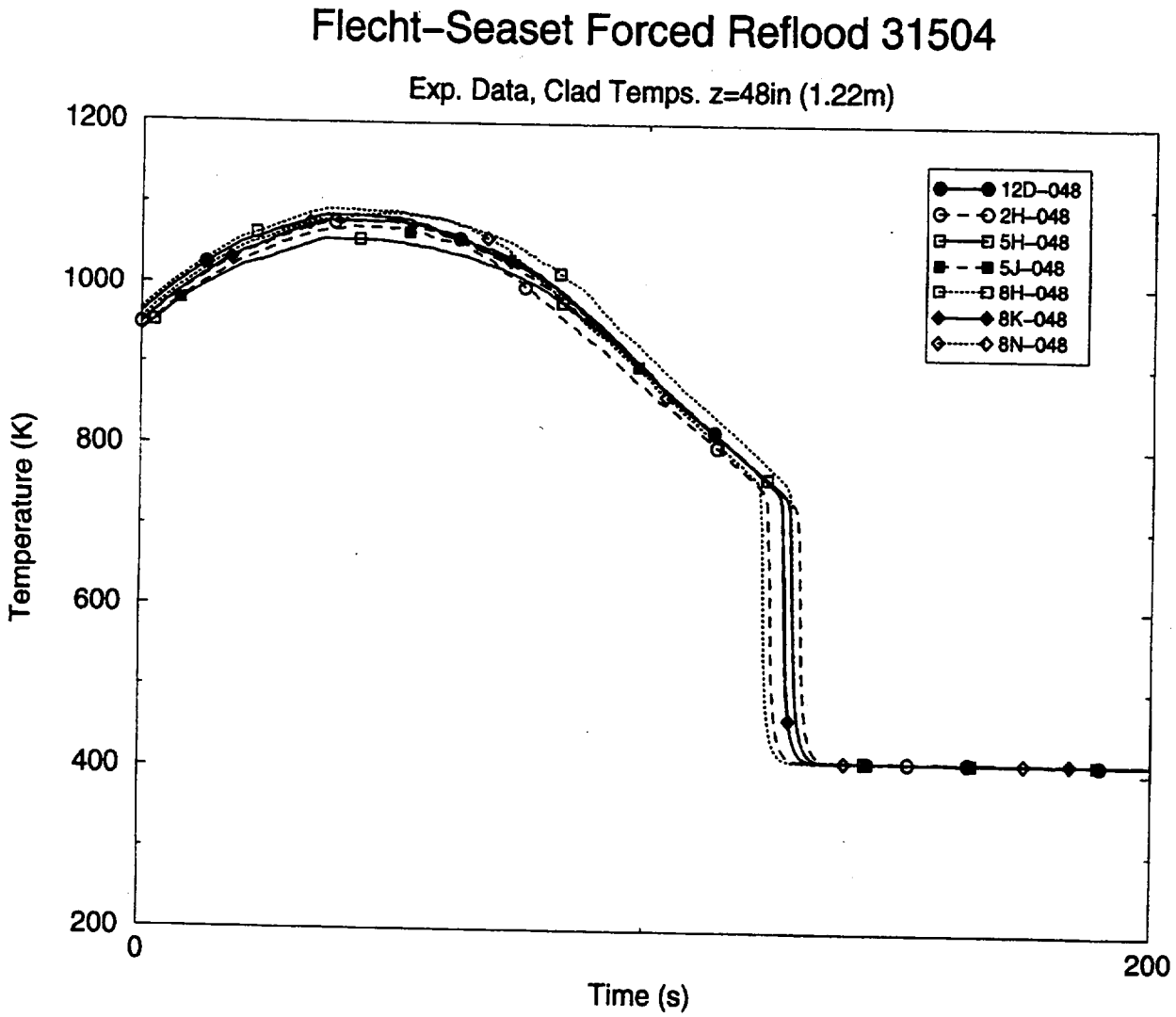



Figure C.1 Flecht-Seaset 31504 Exp. Clad Temperatures, z=1.22m

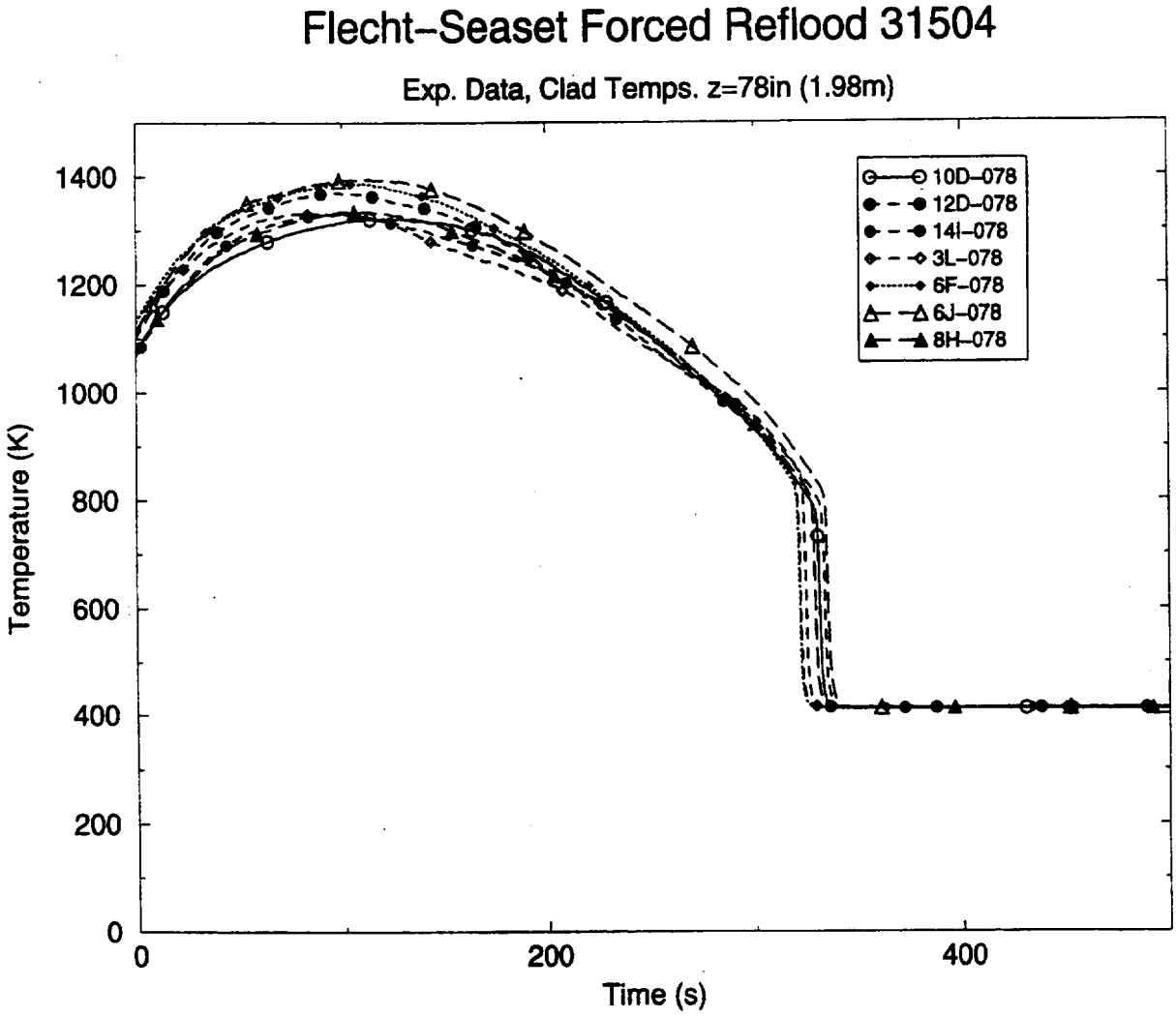



Figure C.2 Flecht-Seaset 31504 Exp. Clad Temperatures, z=1.98m

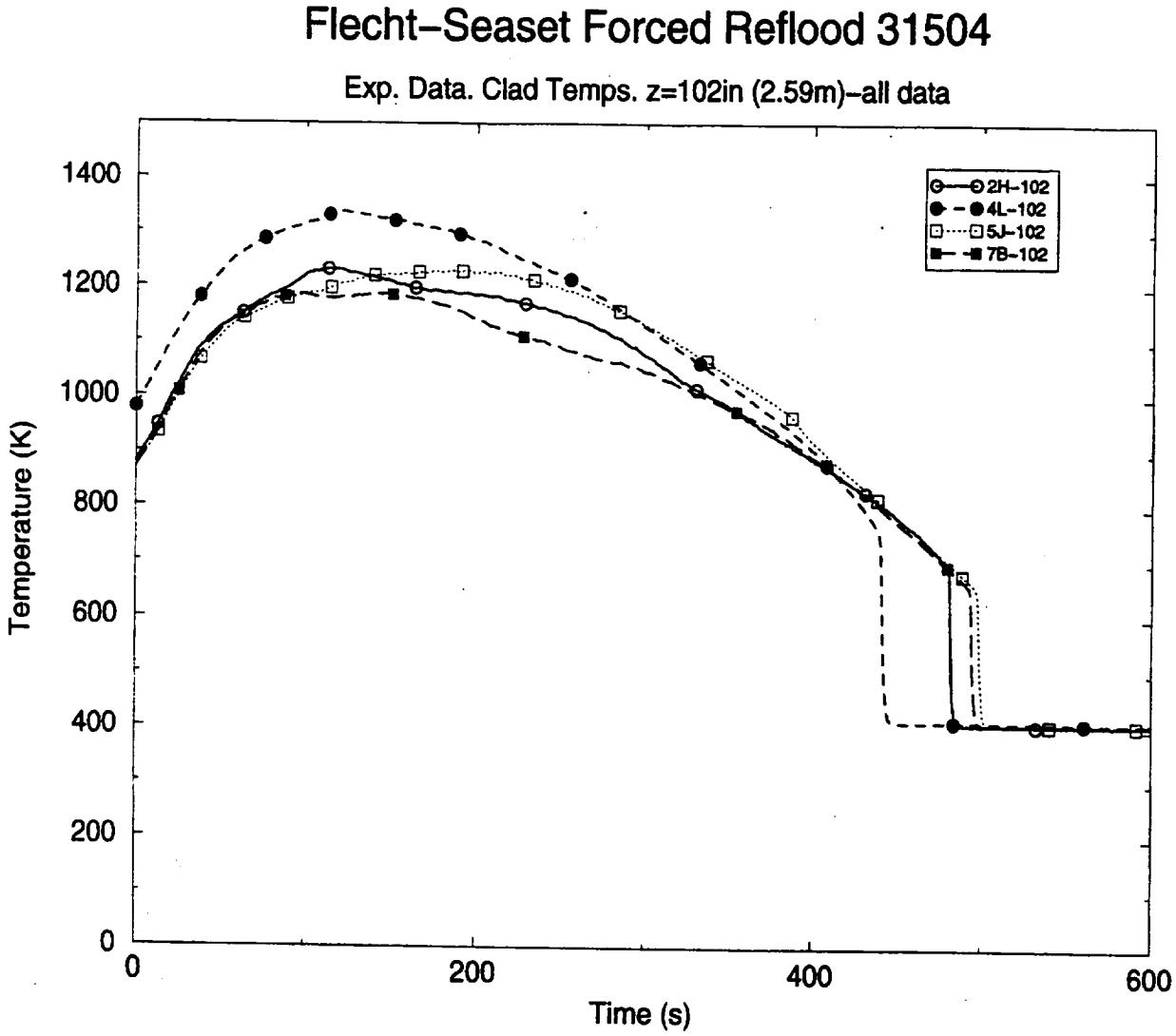



Figure C.3 Flecht-Seaset 31504 Exp.Clad Temperatures, z=2.59m

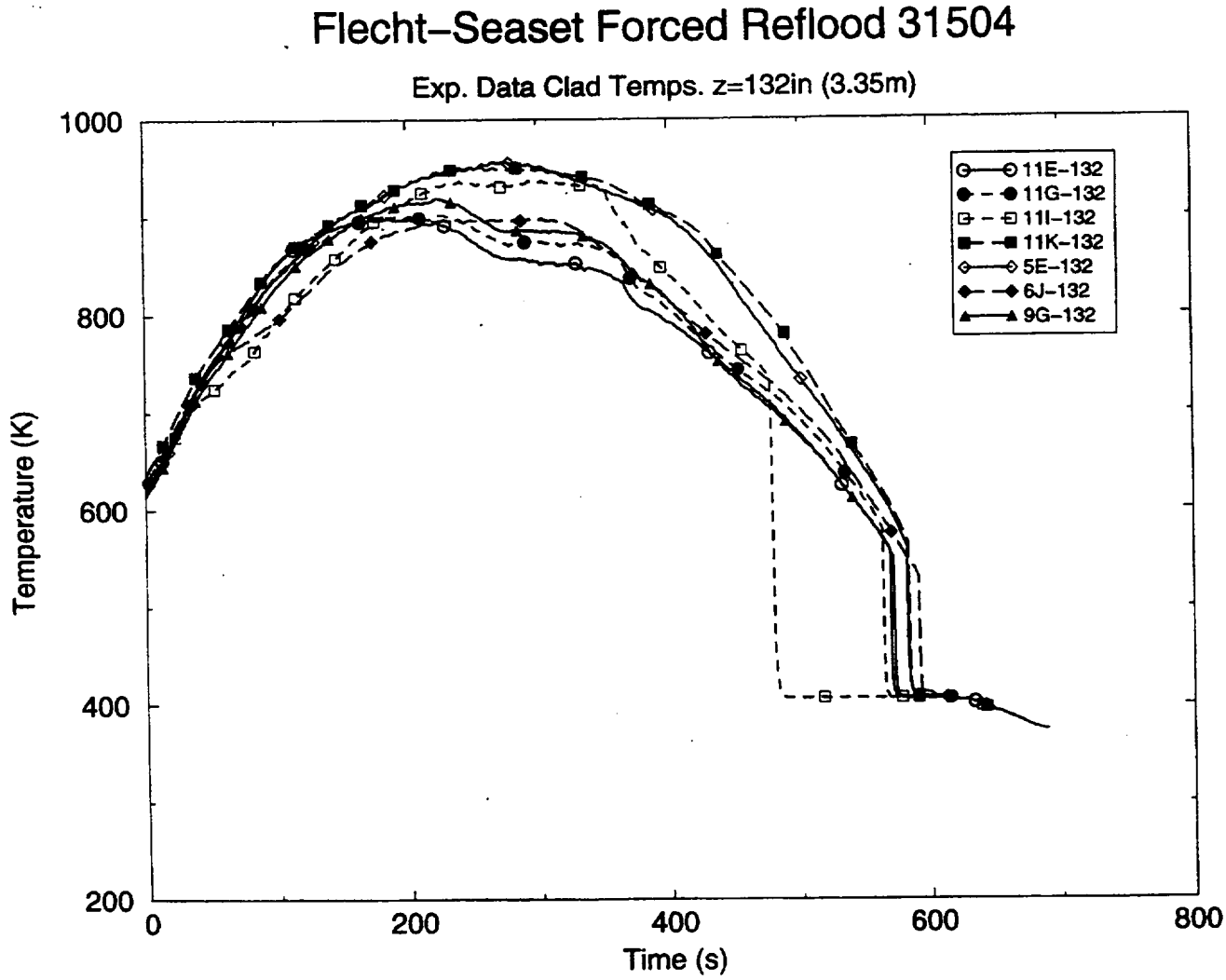



Figure C.4 Flecht-Seaset 31504 Exp. Clad Temperatures, z=3.35m

NRC FORM 335 (2-89) NRCM 1102, 3201, 3202

U.S. NUCLEAR REGULATORY COMMISSION

# BIBLIOGRAPHIC DATA SHEET

*(See instructions on the reverse)*

1. REPORT NUMBER (Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)

NUREG-1737

2. TITLE AND SUBTITLE

Software Quality Assurance Procedures for NRC Thermal Hydraulic Codes

3. DATE REPORT PUBLISHED

| MONTH | YEAR |
|---|---|
| December | 2000 |

4. FIN OR GRANT NUMBER

5. AUTHOR(S)

Frank Odar

6. TYPE OF REPORT

Technical

7. PERIOD COVERED *(Inclusive Dates)*

8. PERFORMING ORGANIZATION - NAME AND ADDRESS *(If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)*

Division of Systems Analysis and Regulatory Effectiveness
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington D.C. 20555-0001

9. SPONSORING ORGANIZATION - NAME AND ADDRESS *(If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)*

Same as above

10. SUPPLEMENTARY NOTES

11. ABSTRACT *(200 words or less)*

This report describes quality assurance procedures for development and maintena nce of the NRC thermal hydraulic codes to be used in reactor plant system transient analysis. These procedures present r equirements for documentation, review, testing and assessment of the thermal hydraulic codes. These procedures will be used b y the NRC staff, its contractors and partners in the code development and maintenance programs.

12. KEY WORDS/DESCRIPTORS *(List words or phrases that will assist researchers in locating the report.)*

SOFTWARE QUALITY ASSURANCE PROCEDURES
THERMAL HYDRAULICS
COMPUTER CODES

13. AVAILABILITY STATEMENT

unlimited

14. SECURITY CLASSIFICATION

*(This Page)*

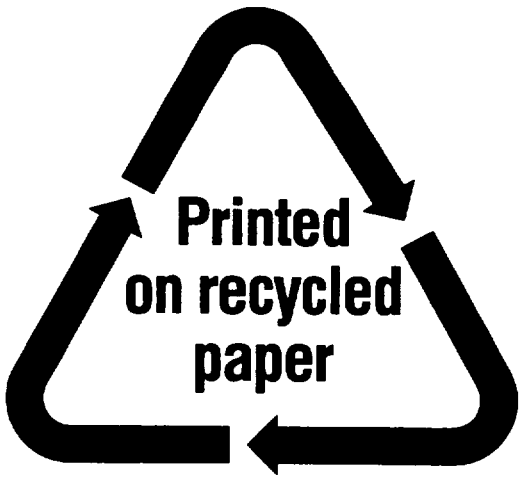
unclassified

*(This Report)*

unclassified

15. NUMBER OF PAGES

16. PRICE




Federal Recycling Program

UNITED STATES
NUCLEAR REGULATORY COMMISSION
WASHINGTON, DC 20555-0001

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300