Title: **TOP–018 DEVELOPMENT AND CONTROL OF SCIENTIFIC AND ENGINEERING SOFTWARE**

EFFECTIVITY AND APPROVAL

Revision 10 of this procedure became effective on August 5, 2005. This procedure consists of the pages and changes listed below.

Page No.	Change	Date Effective
1	3	5/12/2010
2-3	1	6/11/2007
4-5	2	3/13/2009
6-7	1	6/11/2007
8-9	2	3/13/2009
10	1	6/11/2007
11	2	3/13/2009
12-14	0	8/25/2005
15	2	3/13/2009
16	0	8/25/2005
17	3	5/12/2010
18	2	3/13/2009

Change 2: Removes the term source code in favor of more generic terminology.
Change 3: Reflects QA title change

Supersedes Procedure No. TOP–018, Revision 10, Chg 2 dated 3/13/2009.

Prepared by	Date	Approved by	Date
	4/28/2010		4/28/2010

TOP–018 DEVELOPMENT AND CONTROL OF SCIENTIFIC AND ENGINEERING SOFTWARE

1. PURPOSE

The purpose of this procedure is to describe the requirements for scientific and engineering computer software acquisition, use, development, modification, configuration control, and validation. This procedure applies to software used by or for the Geosciences and Engineering Division (Division). Specific controls are established for (i) acquired and (ii) developed and modified software. Controls are applicable to each software version to be used or developed.

This procedure implements the requirements defined in the Division Quality Assurance Manual, Chapter 3.

2. RESPONSIBILITIES

2.1 The cognizant manager is responsible for identifying software subject to development, modification, configuration, and validation controls.

2.2 The software installer is the Information Management Systems Group staff member who is responsible for completing Sections 1-4 of the Software Release Article Acquired Software (form TOP–6-1) in collaboration with the software user. For network software installations, the installer will coordinate with Division Management to identify a user for installation testing.

2.3 The software user is responsible for complying with software use protocols and restrictions.

2.4 Primary users of acquired software are responsible for placing acquired software under control in accordance with Section 4 of this procedure.

2.5 The software developer is responsible for developing (or modifying) and controlling software in accordance with Section 5 of this procedure. The software developer is also responsible for controlling software development files.

2.6 The software custodian is responsible for performing quality assurance (QA) verification activities, maintaining the master directory of software, and supporting the staff and management with proper tools for tracking software and documentation changes.

3. SOFTWARE USE CONTROL

3.1 Identification of Software for Control

Scientific and engineering software used to conduct technical studies shall be controlled in accordance with this procedure. Calculations, including those performed using software controlled in accordance with this procedure, shall be documented and verified according to QAP–014, Documentation and Verification of Scientific and Engineering Calculations.

Software development and use and associated activities shall be identified as described in QAP–013, Quality Planning. Quality planning shall identify software to be used in technical activities (requiring configuration control), and whether the software will be used in regulatory reviews (requiring validation), and developed or modified. The manager and director/assistant director shall jointly determine whether software used in activities other than regulatory reviews needs to be validated.

3.2 Software Exempt from TOP–018 Control

3.2.1 Single-use software specific to a particular application and not intended for distribution does not require control in accordance with this procedure. Calculations using such software shall be documented and verified in accordance with QAP–014. Calculations may be performed using software not yet placed under control of this procedure and may be used in technical products provided that the calculations are documented and verified in accordance with QAP–014. Uncontrolled software should not be used for regulatory reviews.

3.2.2 Commercially available software having built-in computational formulae or programming language, such as Excel, Mathcad, Matlab, Mathematica, GoldSim, and others, do not require control in accordance with this procedure, but calculations using the software shall be documented and verified in accordance with QAP–014. When these types of software are used to develop general-purpose codes or models intended for distribution, the code or models shall be subject to the requirements for developed software described in Section 5 of this procedure.

4. ACQUIRED SOFTWARE CONTROL

4.1 Installation Testing

Prior to configuration control and release, an installation test shall be performed on acquired software for each intended platform and operating system. Installation will be performed by a member of the Information Management Systems Group who is responsible for completing Sections 1-4 of TOP–6–1 in collaboration with a software user designated by the manager. Installation testing of acquired software shall be performed by a software user assigned by

the manager. Installation testing shall consist of running one or more test cases representative of the intended application. Validation tests, when performed prior to placing the software under configuration control, may be used for installation testing. Differences between the installation test results and anticipated results shall be identified and the software shall not be released until the problem is rectified. Installation tests shall be documented in a scientific notebook or the Software Release Notice, TOP–6–1 form. If a scientific notebook documents the installation tests, the software release notice shall identify scientific notebook number and page.

The installation testing is required only for the initial installation of a software version.

4.2 Configuration Control and Release

4.2.1 Software shall be considered to be under configuration control and released for use after the following are provided to the software custodian:

- A completed Software Release Notice, form TOP–6–1.
- A copy of the software (see Section 4.2.2). If copyright protection or licensing prevents copying the software, an explanation will be provided on the software release notice form. Any necessary scripts, make files, relocatable libraries, and executable files shall also be provided.

4.2.2 The software submitted for configuration control shall meet the following criteria:

- Media (tapes, disks, etc.) labels shall contain the name and version of the software.
- Preferred media for all computer files are compact disks. Also acceptable are 8-mm (Exabyte 8200 format) tape cartridge cassettes for larger files. Computer files should be in uncompressed format.

4.2.3 The software custodian shall perform a QA verification to ensure that the requirements of configuration control and release are met. After the verification is completed, the software shall be added to the master list of controlled software that is accessible by the Division staff.

4.3 Software Validation

4.3.1 Validation is performed to gain additional confidence that the software successfully implements the underlying theory and algorithms.

4.3.2 Software that has been approved or endorsed by the U.S. Nuclear Regulatory Commission (NRC) for specific applications may be used in regulatory reviews without validation. In these cases, documentation shall be prepared and submitted to the software custodian, including

- Identification of the NRC document(s) referencing the specific source of the NRC endorsement (i.e., Standard Review Plans, Regulatory Guides, etc.)

- Listing of the NRC approved applications of the software

- Any limitations on use of the software

Software endorsed by NRC shall be subject to the configuration controls described in Section 4.2. When using software endorsed by NRC, the Division will retain responsibility for using it in a technically acceptable manner. The master list of controlled software shall indicate such endorsement.

4.3.3 A software validation plan shall be prepared to describe and justify the approach to be taken for validation. A template of a software validation plan providing detailed content and format requirements shall be available on the Division computer network. The plan shall describe

- Tests that provide evidence of correct and successful implementation of the functions and/or benchmarking or comparative testing against results from other software. As appropriate, existing test cases may be used for acquired software (see Section 4.3.5).

- When available and applicable, comparisons of test results to research papers, laboratory experiments, existing test cases or previously published sources of data and industry experience in using the software in comparable applications (see Section 4.3.4).

The scope of validation shall be commensurate with anticipated software use. Validation may be limited to a range of the full capability of the software. The software validation plan and validation report shall clearly indicate whether the validation covers all functions (i.e., full) or, if limited, shall identify the software functions to be validated.

4.3.4 The extent and balance of validation methods (i.e., testing, benchmarking, examples of prior industry use, and published work) shall be sufficient to establish confidence in the software for its intended application. Software with well-documented use in comparable applications may not need extensive testing to establish confidence in its use.

4.3.5 The test case(s) shall demonstrate that the software functions are properly and correctly implemented. This demonstration may consist of comparisons with known solutions. In special cases where known or analytical solutions are not available, the computational demonstration may be accomplished by direct hand calculation or comparison to computations obtained with commercial mathematical software such as MathCad,

Mathematica, or Matlab. Each test case should have a clear and concise description of the functions involved, method(s) of testing, and expected results.

4.3.6 The software validation plan may include benchmarking or comparisons with results from other software. Each benchmark case should have a clear and concise description of the problem definition, goal(s) of the test, method(s) of testing, and expected results. Previously published sources of benchmark data may be utilized that show adequate comparisons between the software to be validated and other similar software. If so, these previously published sources shall be identified in the software validation plan.

4.3.7 The software validation plan shall be reviewed in accordance with QAP–002, Review of Documents, Reports, and Papers, to verify that the planned software validation is sufficient and appropriate.

4.3.8 Test case and benchmarking results may take the form of hard-copy output of graphical and/or tabular results. The software user shall evaluate the results based on the submitted tabular and/or graphical comparisons.

4.3.9 A validation report shall be prepared by the software user documenting the results of the validation cases, interpretation of the results, and identification and resolution of any problems. The report will also document the degree of the software validation (e.g., limited or full) and any limitations identified in validated functions (see Section 4.3.3). The report shall be reviewed in accordance with QAP–002 to verify that the results and comparisons adequately support software validation.

For a relatively simple validation, the software validation plan and validation report may be combined in a single document and reviewed in accordance with QAP–002.

4.3.10 If the scope of the validation is increased over a previous validation, the software validation plan and validation report shall be revised to reflect the increased scope.

4.4 Software Retirement

Software should be removed from control and uninstalled from computers when it is no longer used or subject to the requirements of this procedure. The basis for retirement may include

- Availability of more appropriate software
- Changes in regulatory requirements that make the software no longer applicable
- Completion of the project/program using the software

A software release notice shall be issued to document the retirement of software. Retired software shall be removed from the master list of controlled software. Archives of retired software shall be maintained.

5. DEVELOPED AND MODIFIED SOFTWARE CONTROL

5.1 Software development and modification shall be accomplished following the procedure defined in this section. At a minimum, initial software development shall be based on a software requirements description (Section 5.2) and a software development plan (Section 5.3). Subsequent modifications to software with a previously approved software requirements description and software development plan may not require revisions to these documents if the scope or validity of these documents is not affected (see Sections 5.2.4 and 5.3.3).

5.2 Software Requirements Identification

5.2.1 A template of the software requirements description that provides detailed content and format requirements shall be maintained and available on the Division computer network.

5.2.2 Proof-of-concept development and testing of specific software components prior to preparing a software requirements description may be permitted with manager approval. This authorization shall be documented and maintained in the software development file.

5.2.3 The software requirements description and subsequent revisions shall be reviewed in accordance with QAP–002. The approved software requirements description shall be provided to the software custodian as QA records.

5.2.4 When substantive changes are planned to the objective, mathematical basis or numerical algorithm, the software requirements description shall be revised. A software change report (see Section 5.9) shall be used to specify modifications to existing software when revision of the software requirements description is not necessary.

5.3 Software Development Planning

5.3.1 A template providing detailed content and format requirements for the software development plan shall be maintained and available on the Division computer network. The software development plan shall describe the activities necessary to develop or modify the software, the standards and testing that will be applied to the software, an estimate of the work to be performed, the necessary resources, and a proposed schedule.

Relatively small development efforts may not benefit from the same level of planning that is appropriate for large efforts. In such cases, the software requirements description and software development plan may be combined into a single document.

5.3.2 The software development plan shall be reviewed in accordance with QAP–002. Division QA shall verify the plan is properly implemented through surveillance.

5.3.3 When substantive changes are planned to the developmental approach, the software development plan shall be revised.

5.4 Software Development

5.4.1 Software development, including informal testing of modules and subroutines, shall be documented in software development files. Development documentation shall be sufficient so that an informed reviewer can follow the procedures and logic of software development.

5.4.2 Software development should follow the coding guidelines identified in the software development plan [e.g., Cook, et al. (1990) for FORTRAN codes and Cook, et al. (1991) for C language codes]. These guidelines should be applied to the modified portions of codes as well. For modifications to acquired software, the modifications should follow the coding conventions of the acquired software to maintain consistency in style.

5.4.3 Software shall be documented internally to allow a user to understand the function(s) being performed and to follow the execution flow. Recommended guidelines include

- Define the function(s) performed by the software or module.
- Define the purpose, units (e.g., meters, second, kg, etc.), and possible range of each data element or collection of data elements.
- For modified software, provide a comment that identifies the software change report that documents the basis for the change or group of changes.
- Include as appropriate, other design details (e.g., trade-off decisions, general discussion of design algorithms, etc.) in related help files and/or scientific notebook entries. On occasion, internal documentation may be appropriate on a line-by-line basis.

5.4.4 The title block shall include, as applicable, the program title, developed for (customer), contract number and other relevant customer information, date, customer contact(s) and phone number, and software developer and phone number. Disclaimer and copyright notices shall be included in the main program header, if appropriate.

5.4.5 The header information shall provide (i) program name; (ii) customer name, (iii) revision history; and (iv) software requirements description section number and/or software change report number of the requirement(s) being implemented. All modules within a program shall follow the same basic header conventions. If a particular piece of information is not applicable, then the information portion should be filled in with an N/A.

5.4.6 Software shall be designed so output files are uniquely identified by date, time, name (title), and version. Individual runs shall also include any appropriate disclaimer notices.

5.4.7 Code reviews of software modules shall be performed as and when described in the software development plan. Informal code reviews are not required to be documented.

Code reviews, when determined necessary, shall

- Verify compliance with standards file headers, variable names, and comments.
- Identify potential problems, omissions, and contradictions.
- Identify ways to improve the code; and consider alternative implementations.
- Achieve greater uniformity in technical work.
- Confirm those parts of the software in which improvement is either not desired or not needed.

Code reviews, should be performed early in the development life cycle to maximize detection of potential defects and reduce the risk of unnecessary rework and testing. Code review results shall be documented in a scientific notebook or software development files that allow others to understand the code review process. A defect log may be generated to identify the defect, reviewer, and recorder.

5.5 User's Manual

5.5.1 The user's manual contains basic instructions for software use and shall be prepared when required by the software development plan or by client direction. The user's manual may be embedded in the software or may be issued as a separate document. The user's manual may be submitted in preliminary form at the discretion of the manager. If a preliminary manual is submitted prior to software release, a date for submission of the final version will be presented.

5.5.2 The User's Manual shall provide

- A clear and concise explanation of the purpose of the software.
- Step-by-step instructions for the installation and use of the software.
- A description of computer resources required for installation and operation (i.e., hardware platform, operating system, ANSI language version, special compiler instructions, list of necessary computer files).
- Statements of the capabilities and limitations of the software. Test cases should be included that, when executed during initial installation, will assure the user the software will operate as intended.

5.5.3 The user's manual should, to the extent practical, be written in accordance with the guidelines for software documentation in NUREG–0856 (NRC, 1983) and NUREG/BR–0167 (NRC, 1993). A user's manual may be documented (i) electronically (i.e., a readme file, help file), (ii) in hard copy, or (iii) as a combination of electronic and hard copy.

5.5.4 The user's manual shall be reviewed as a technical report in accordance with QAP–002.

5.6 Installation and Acceptance Testing

5.6.1 An installation test of software shall be performed on each intended platform and operating system by the software developer prior to controlled release.

One or more test cases representative of the intended application shall be run. If software validation (Section 5.8) is completed prior to release, validation test cases may be used for installation testing. Differences between the test run results and the anticipated results shall be identified. If the differences are beyond what may be reasonably expected, the software shall not be released until the problem is rectified. Installation tests shall be documented in the software development files or the Software Release Notice.

5.6.2 Acceptance testing demonstrates whether the requirements for developed or modified software specified in the software requirements description/software change report have been fulfilled. Acceptance testing shall be performed to ensure that the software meets specified requirements.

Acceptance testing shall be conducted for each intended computer platform and operating system. Acceptance testing shall be documented sufficiently that an informed reviewer can follow the procedures and logic of acceptance testing. The test procedures shall describe the requirement being tested, the modules/units being tested, the test conditions, a list of the inputs, a list of outputs, the expected results, any procedures used, and the equipment and facilities needed to perform the test. Test cases, including input and output files, shall be documented to facilitate testing future modifications to the software.

Acceptance testing shall be documented in a scientific notebook, a stand-alone report, or in a software change report (see Section 5.9). If a formal stand-alone report is prepared for delivery to a client, it shall be reviewed in accordance with QAP–002. After acceptance testing is successfully completed, the software may be placed under configuration control and released for use.

5.7 Configuration Control and Release

5.7.1 Developed and modified software shall be placed under configuration control (i.e., controlled and released) prior to delivery to the user.

5.7.2 A preliminary version (i.e., beta version) of Division-developed software may be released to obtain user feedback. Beta release software should be customer usable and have essential features present. A beta version is not fully tested and therefore may not be free of bugs. A beta release shall include a disclaimer stating that the software is issued to solicit user feedback and shall not be used for regulatory review or client-specific activities. This disclaimer may be placed on the CD, in the program title block, in the "About file," or other conspicuous location.

To issue a beta release, the software developer shall provide (i) a copy of the software and (ii) a software release notice to the software custodian.

5.7.3 The software developer shall provide the following to the software custodian for software to be controlled and released:

- A copy of the software
- A completed software release notice, form TOP–6–1
- Evidence of acceptance and installation testing (see Section 5.6)

5.7.4 When applicable, the software release notice for modified software shall indicate any expected deviations from past verification and benchmark data that are likely to be caused by changes to software, subroutines, or functions.

5.7.5 The software medium submitted for software configuration control shall meet the following criteria:

- Medium (tapes, disks, etc.) labels shall contain the name and version of the software.
- Preferred media for all computer files are compact disks. Also acceptable is 8-mm (Exabyte 8200 format) tape cartridge cassettes for larger files. Computer files should be in uncompressed format.

5.7.6 The software custodian shall perform QA verification that applicable requirements are met. Upon release of software, the software shall be added to the master list of controlled software that is accessible by the Division staff.

5.8 Software Validation

5.8.1 Validation is performed to gain additional confidence that the software successfully implements specified requirements.

5.8.2 A software validation plan shall be prepared to describe and justify the approach to be taken for validation. A template of a software validation plan providing detailed content and format requirements shall be available on the Division computer network. The plan shall describe

- Tests that provide evidence of correct and successful implementation of the underlying theory and algorithms and/or benchmarking or comparative testing against results from other software.

- When available and applicable, comparisons of test results to research papers, laboratory experiments, existing test cases, or previously published sources of data.

The scope of validation shall be commensurate with anticipated software use. Validation may be limited to a range within the full capability of the software. The software validation plan and validation report shall clearly indicate whether the validation covers all functions (i.e., full) or, if limited, shall identify the software functions to be validated.

As applicable, the software validation plan shall describe the modules to be validated and the couplings (if any) between modules. For software with separately operable modules (e.g., PCSA), the software validation shall describe which modules or functions are to be validated and the couplings (if any) between modules.

5.8.3 The extent and balance of validation methods (i.e., testing and benchmarking) shall be sufficient to establish confidence in the software for its intended application.

5.8.4 The test case(s) shall demonstrate that the functions of the software are properly and correctly implemented. This demonstration may consist of comparisons with known solutions. In special cases where known or analytical solutions are not available, the computational demonstration may be accomplished by direct hand calculation or comparison to computations obtained with commercial mathematical software such as MathCad, Mathematica, or Matlab. Each test case should have a clear and concise description of the functions involved, method(s) of testing, and expected results.

5.8.5 The software validation plan may include benchmarking or comparisons with results from other software. Each benchmark case should have a clear and concise description of the problem definition, goal(s) of the test, method(s) of testing, and expected results. Previously published sources of benchmark data may be utilized that show adequate comparisons between the software to be validated and other similar software. If such exists these previously published sources shall be identified in the software validation plan.

5.8.6 The software validation plan shall be reviewed in accordance with QAP–002 to verify that the planned software validation is sufficient and appropriate.

5.8.7 Test case and benchmarking results may take the form of hard-copy output of graphical and/or tabular results. The software developer shall evaluate the results based on the submitted tabular and/or graphical comparisons.

5.8.8 A validation report shall be prepared by the software developer/user documenting the results of the validation cases, interpretation of the results, and identification and resolution of any problems. The report also will document the degree of the software validation (e.g., limited or full) and any limitations identified in validated functions. The report shall be reviewed in accordance with QAP–002 to verify that the results and comparisons adequately support software validation.

For a relatively simple validation, the software validation plan and validation report may be combined in a single document and reviewed in accordance with QAP–002.

5.8.9 When validated software is changed or the scope of the validation is expanded, the software validation plan and validation report shall be revised to reflect the change. A software change report may be used for minor modifications when revision of the plan or report are not warranted.

At a minimum, regression testing shall be performed to determine whether the changes affected the previously validated functions.

5.9 Software Change

5.9.1 Operational problems shall be reported and resolved, and planned changes to released software, including changes to the computer platform or operating system, shall be accomplished in accordance with the software development requirements of Sections 5.1–5.7. Changes to validated software shall be subject to revalidation in accordance with Section 5.8.

5.9.2 The software requirements description and the software development plan shall be revised if the software changes affect the scope and/or accuracy of these documents. For less significant changes, the software change report may be used to document the changes and acceptance testing.

5.10 Software Retirement

Software should be removed from control and uninstalled from computers when it is no longer needed or subject to the requirements of this procedure. The basis for retirement may include

- Availability of more appropriate software
- Changes in regulatory requirements that make the software no longer applicable
- Completion of the project/program using the software

A software release notice shall be issued to document the retirement of software, and the retired software shall be removed from the master list of controlled software. Archives of retired software shall be maintained.

6. RECORDS

6.1 The following documents shall be controlled as QA records in accordance with QAP–012, "Quality Assurance Records Control":

- Software requirements description
- Software development plan
- Software release notice
- User's manual
- Software change report
- Software validation plan and validation report
- Software development files (including installation and acceptance test results)
- Controlled software listings

6.2 A copy of the released software on appropriate media shall be maintained as a QA record in accordance with QAP–012.

7. REFERENCE DOCUMENTS

Cook, D.M., G.V. Miller, J.R. Olsen, D.M. Snider, and K.L. Wagner. *C Language Coding Practices.* EGG–CATT–9812. Idaho Falls, Idaho: EG&G Idaho, Inc. 1991.

Cook, D.M., N.H. Marshall, E.S. Marwil, S.D. Mathews, and G.A. Mortenson. *FORTRAN Coding Guidelines.* EGG–CATT–8898. Idaho Falls, Idaho: EG&G Idaho, Inc. 1990.

NRC. NUREG/BR–0167, "Software Quality Assurance Program and Guidelines." Washington, DC: NRC. February 1993.

———. NUREG–0856, "Final Technical Position on Documentation of Computer Codes for High-Level Waste Management." Washington, DC: NRC. June 1983.

8. DEFINITIONS

Acceptance Testing: Formal testing that is conducted to determine whether the software satisfies the requirements described in the software requirements description.

Acquired Software: Software developed by an individual or organization other than the Division while not employed by or under contract to the Division to develop that software.

Benchmarking: A process of testing the simulation capability of software by making comparisons against output of other software or other independent analyses.

Beta Release: The software product is ready to be appraised by outsiders, but software product development is not complete. Typically, the core software product is complete, and all essential features described in the software requirements description or software change report(s) are present and tested.

Code Review: An in-depth review of a software program or a portion of a software program by qualified personnel who are independent of those who performed the work.

Commercial-off-the-shelf: Acquired software products that are ready-made and available for sale to the general public.

Computer Code: A set of computer instructions for performing operations specified in a numerical model or other computer program.

Configured Software: Software of a fixed configuration and identification that has met the appropriate criteria for configuration control.

Configuration Control: The procedure for (i) baselining configurations of software, (ii) controlling changes to software, and (iii) releasing configured software to users. Configuration control is implemented at two levels: developmental and delivered. Configuration control at the developmental level enables the software developer to track software changes and problems to ensure the integrity of the software. Configuration control at the delivered level enables the software custodian to track baseline items and changes thereto, specifically those that are listed as deliverables to the customer.

Developed Software: Software that has been developed or modified by the Division or by an individual or organization employed by or under contract to the Division. Developed software is controlled in accordance with the pertinent provisions of TOP–018.

Existing Software: Software that has been acquired or developed by CNWRA outside the controls provided by TOP–018. Typically, codes in this category were acquired or developed before the procedure was implemented and have since been superseded by other codes or later versions of the same code. The Division management has decided that existing software will not be brought into compliance with the provisions of this procedure. Such codes cannot be used in support of NRC licensing actions or regulatory reviews.

Installation Testing: A functional test in which a test case(s) representative of the intended software application is run to determine if the software has been properly installed on the target computer system.

Mathematical Model: A mathematical representation of processes or phenomena.

Model Validation: Process of assuring that a mathematical model as embodied in a computer code (software) is an adequate representation of the process or phenomenon for the purpose for which it is intended.

Modified Software: Software altered in accordance with this procedure. Modified software may have originally been acquired or developed.

Module: A component of a computer code. A module may be one or a group of subroutines or functions.

Not to be Modified: This term applies only to acquired software, and affects the provisions of TOP–018 that are pertinent to the code.

Numerical Method: A procedure for solving (or approximately solving) a problem primarily by a sequence of arithmetic operations.

Program: A list of instructions, written in a programming language, that a computer can execute so that the machine acts in a predetermined way.

Prototype: A preliminary version of the software components that may be developed for scoping and testing purposes.

Quality Assurance Verification: The process of evaluating software and associated documentation to determine whether the software and associated documentation satisfies applicable procedural requirements.

Regression Testing: Testing that is performed after making a change to a software program or environment (e.g., operating system). Its purpose is to determine if the change has introduced new errors into the software. Regression testing is usually performed by running some subset of the software acceptance, installation, and/or validation test cases, then checking the test results with a baseline of correct answers.

Regulatory Review: Encompasses all activities conducted in direct support of NRC fulfilling its licensing or certifying roles under applicable regulations at Title 10 of the Code of Federal Regulations. Regulatory review includes, but is not limited to document review; independent numerical, sensitivity, and importance analyses and the like; check calculations; confirmatory tests or measurements in the laboratory and the field; evaluations of compliance with regulatory requirements; and preparation of safety evaluation reports, environmental documentation, and the like.

Risk Management: A process of identifying risks and mitigating their effects before the risks have disrupted project activities. Risks may adversely affect project schedule, cost, software performance, or other factors. Risk management includes risk identification, analysis, prioritization, and control.

Software Change Report: A document prepared by the software developer/user to identify and correct problems or make other minor changes in configured software or software documentation.

Software Custodian: QA staff, independent of the software developers and users responsible for controlling the QA software files and documentation and maintaining the configuration control system.

Software Developer: The original author or a person who develops or modifies software.

Software Development Files: A physical location for material pertinent to the development of a particular body of software. Used on all software development projects, software development files may be maintained as hard copy folders, loose-leaf notebooks or scientific notebook, electronic files, or a combination of hard copy and electronic files. The software development files shall be identified with the appropriate software project number, identify the software module(s) associated with the software development file, and be maintained by the software developer.

Software Development Plan: A document that describes the project plans for conducting a development effort. The software development plan covers new development, modification, maintenance, and all other activities resulting in software products and documents. It provides a tool for monitoring software development, including the methods to be used and the approach to be followed for each activity. The software development plan includes plans for software configuration management and risk management.

Software Requirements Description: A document that describes the technical bases for software development and provides a vehicle for client approval of software development or changes.

Software User: The organization, person, or persons who will use the software product being developed, acquired software, or are involved in the application of software covered by this procedure.

Software Validation: Provides evidence of correct and successful implementation of the underlying theory and algorithms as outlined in the software requirements description, software change report, or other relevant documentation, such as user's manuals for acquired software. Software validation is a software testing process that is distinct from model validation (defined earlier).

Subroutine: A set of instructions with well-defined inputs and outputs that is used by modules in a program.

Target System: Hardware platform and operating system designated for the intended use of the code.