NOT MEASUREMENT SENSITIVE

DOE G 414.1-4A
XX-XX-20XX
1-30-15 Draft

SOFTWARE QUALITY ASSURANCE GUIDE for Use with DOE O 414.1D, QUALITY ASSURANCE

[This Guide describes suggested nonmandatory approaches for meeting requirements. Guides are not requirements documents and are not construed as requirements in any audit or appraisal for compliance with the parent Policy, Order, Notice, or Manual.]

U.S. DEPARTMENT OF ENERGY
Washington, D.C.

DISTRIBUTION:
www.directives.doe.gov

INITIATED BY:
Office of Environment, Health, Safety and Security

FOREWORD

This Department of Energy (DOE) Guide is approved for use by all DOE organizations, including the National Nuclear Security Administration (NNSA). The Guide was developed in support of DOE Order (O) 414.1D, *Quality Assurance*, dated 4-25-2011 (Change Notice No.1, dated May 2013). The Guide assists DOE and its contractors with information and guidance on the quality assurance (QA) requirements of DOE O 414.1D as applied to software used in nuclear and non-nuclear facilities.

This Guide (G) revises and supersedes DOE G 414.1-4, *Safety Software Guide for Use with 10 CFR Part 830, Subpart A, Quality Assurance Requirements and DOE O 414.1C, Quality Assurance*, dated 6-17-2005. The revised Guide addresses a broader range of software (safety and other software) compared to the 2005 version and also provides guidance on the application of the ten QA criteria and on other special topics.

DOE Guides, part of the DOE Directives System, are issued to provide supplemental information regarding DOE's expectations for meeting requirements in rules, orders, notices, and regulatory standards. Guides may also provide acceptable methods for implementing these requirements. Guides are not substitutes for requirements, nor do they replace technical standards used to describe established practices and procedures for implementing requirements. This Guide should not be interpreted as requirements in any audits or appraisals for compliance with associated rules or directives.

Comments, including recommendations for additions, modifications, or deletions, and other pertinent information, should be sent to:

U.S. Department of Energy
Office of Quality Assurance (AU-33)
1000 Independence Avenue SW
Washington, DC 20585-0270

Phone:	301-903-6571
Fax:	301-903-6172
E-mail:	subir.sen@hq.doe.gov

This Guide is available electronically on the DOE Directives System at the following address: http://www.directives.doe.gov.

CONTENTS

1. INTRODUCTION ... 1
1.1 PURPOSE ... 1
1.2 SCOPE ... 1
1.3 RESPONSIBILITY FOR SAFETY SOFTWARE ... 2
1.4 SOFTWARE QUALITY ASSURANCE REQUIREMENTS ... 2
1.5 SOFTWARE QUALITY ASSURANCE PROGRAM ... 3
2. SOFTWARE TYPES, CATEGORIES AND GRADING ... 4
2.1 SOFTWARE TYPES ... 4
2.1.1 Acquired Software ... 4
2.1.2 Configurable Software ... 4
2.1.3 Custom-Developed Software ... 4
2.2 SOFTWARE CATEGORIES AND GRADING ... 5
2.2.1 Safety Software ... 5
2.2.2 OSW ... 8
3. SOFTWARE MANAGEMENT ... 12
3.1 SAFETY SOFTWARE IDENTIFICATION AND MANAGEMENT ... 12
3.1.1 Role of the Design Authority in Safety Software Identification ... 12
3.1.2. Safety Software Inventory Management ... 13
3.1.3 Management of Central Registry and Toolbox Codes ... 13
3.1.4 Legacy Software Management ... 14
3.2 SOFTWARE QUALITY ASSURANCE ELEMENTS ... 15
3.2.1 Use of National/International Consensus Standards ... 15
3.2.2 Continuous Improvement, Measurement, and Metrics ... 15
3.2.3 Software Reliability ... 16
3.2.4 System, Software and Risk ... 16
4. MANAGEMENT OF SOFTWARE BY TEN QA CRITERIA ... 18
5 SELECTING AND IMPLEMENTING SOFTWARE QUALITY ASSURANCE WORK ACTIVITIES ... 19
5.1 OVERVIEW OF SOFTWARE-RELATED WORK ACTIVITIES ... 19
5.2 DETAILED GUIDANCE ON SOFTWARE-RELATED WORK ACTIVITIES ... 20
5.2.1 Software Project Management and Quality Planning ... 21

5.2.2 Software Risk Management 23
5.2.3 Software Configuration Management 25
5.2.4 Procurement and Supplier Management 26
5.2.5 Software Requirements Identification and Management 27
5.2.6 Software Design and Implementation 29
5.2.7 Software Safety Analysis and Safety Design Methods 33
5.2.8 Software Verification and Validation 35
5.2.9 Problem Reporting and Corrective Action 39
5.2.10 Training of Personnel in the Design, Development, Use, and Evaluation of Software 40

6. SPECIAL TOPICS SUPPORTING SOFTWARE QUALITY ASSURANCE 41

6.1 MODEL DEVELOPMENT AND EVALUATION 41
6.1.1 Introduction 41
6.1.2 Model Development 42
6.1.3 Model Validation 43
6.1.4 Model Selection and Evaluation 47
6.2 DIGITAL SYSTEM SOFTWARE QUALITY ASSURANCE 49
6.2.1 Digital System Identification 49
6.2.2 SQA for Digital System Design 50
6.2.3 SQA for Digital System Acquisition 51
6.2.4 SQA for Digital System Use and Maintenance 52
6.3 COMMERCIAL GRADE DEDICATION OF COMPUTER PROGRAMS 52
6.3.1 Introduction 52
6.3.2 Approach 53
6.3.3 Dedication of Commercial Grade Computer Programs 53
6.3.4 Technical Evaluation and Acceptance 54
6.4 SOFTWARE SECURITY ASSURANCE 56
6.4.1 Introduction 56
6.4.2 Methodology for Designing and Implementing Secure Software 57
6.4.3 Testing and Analysis of Software for Security Vulnerabilities 59
6.4.4 Security Controls 60

7. ASSESSMENT 63

7.1 GENERAL 63

7.2 DOE AND CONTRACTOR ASSESSMENTS 63

APPENDIX A. MANAGEMENT OF SOFTWARE BY TEN QA CRITERIA A-1

APPENDIX B. PROCEDURE FOR ADDING, REVISING OR DELETING SOFTWARE IN THE DOE SAFETY SOFTWARE CENTRAL REGISTRY B-1

APPENDIX C. SELECTING AND IMPLEMENTING APPLICABLE WORK ACTIVITIES C-1

APPENDIX D. QUALITY ASSURANCE STANDARDS FOR SOFTWARE D-1

APPENDIX E. SAFETY SOFTWARE CRITERIA REVIEW AND APPROACH DOCUMENT (CRAD) E-1

APPENDIX F. ACRONYMS AND DEFINITIONS F-1

APPENDIX G. REFERENCES G-11

1. INTRODUCTION

1.1 PURPOSE

This Department of Energy (DOE) Guide provides information and acceptable methods for implementing the software quality assurance (SQA) requirements of DOE Order (O) 414.1D, *Quality Assurance*, dated 4-25 2011 (Change Notice No. 1, dated May 2013) for software applications used in nuclear and non-nuclear facilities.

This Guide (G) revises and supersedes DOE G 414.1-4, *Safety Software Guide for Use with 10 CFR Part 830, Subpart A, Quality Assurance Requirements and DOE O 414.1C, Quality Assurance*, dated 6-17-2005. The 2005 version of the Guide only addressed software used in nuclear safety applications; the revised Guide also addresses other software (OSW) which has safety (other than nuclear), mission critical, research applications or general applications. Furthermore this revision significantly updates the previous version in the areas of software category and grading determination and provides additional guidance on the management of software by the ten quality assurance criteria as well as other special topics supporting software quality assurance.

This Guide utilizes generally accepted SQA concepts and practices drawn from governmental and industry consensus standards, and describes their application for software engineering activities. Alternative methods to those described in this Guide may be used, provided they comply with the requirements of DOE O 414.1D.

1.2 SCOPE

The scope of this Guide includes information and methods for the application of quality assurance (QA) for software used in nuclear safety applications and OSW applications.

This guide provides:

- A high level description of (1) responsibilities for SQA, (2) DOE SQA requirements, and (3) performance of SQA under a DOE approved QAP. (Sections 1.3, 1.4, and 1.5)
- Guidance on identification of software types, categories and grading levels (Section 2)
- General information and guidance on software management (Section 3)
- Overview of the management of software by ten QA criteria (Section 4)
- Guidance on selection and implementation of SQA work activities (Section 5)
- Guidance on specialized topics including (1) Model Development and Evaluation, (2) Digital System Software Quality Assurance, (3) Commercial Grade Dedication of Computer Programs, and (4) Software Security Assurance. (Section 6)
- Guidance on Assessments (Section 7)

The appendices provide supporting information and examples on:

- Management of Software by Ten QA Criteria (Appendix A)
- Procedures for Adding, Revising or Deleting Software in the DOE Safety Software Central Registry (Appendix B)
- Selecting and Implementing Applicable Work activities (Appendix C)
- Quality Assurance Standards for Software in DOE Facilities(Appendix D)
- Safety Software Criteria Review and Approach Document (Appendix E)
- Acronyms and Definitions (Appendix F)
- References (Appendix G)

1.3 RESPONSIBILITY FOR SAFETY SOFTWARE

The Office of Environment, Health, Safety and Security (EHSS), previously Office of Health, Safety and Security, has the lead responsibility for developing requirements and guidance through the DOE Directives System for safety software per DOE O 414.1D.

The organizations using safety software should determine whether to qualify a software item for safety applications (see Section 2.2.1.4). Organizations should also coordinate with the Office of the Chief Information Officer regarding corporate SQA procedures. DOE line organizations are responsible for providing direction and oversight of the contractor’s implementation of SQA requirements.

1.4 SOFTWARE QUALITY ASSURANCE REQUIREMENTS

DOE O 414.1D requirements supplement the quality assurance program (QAP) requirements of Title 10 Code of Federal Regulations (CFR) 830, Subpart A, *Quality Assurance* (hereafter QA Rule). The QA Rule establishes the “quality assurance requirements for contractors conducting activities, including providing items or services that affect, or may affect, nuclear safety of DOE nuclear facilities.” DOE O 414.1 D supplements the QA Rule by identifying software as an item subject to the ten QA criteria in the QA Rule. In DOE O 414.1D, Requirements (paragraph 4. a. (2)) and Attachment 1, Contractor Requirements Document (paragraph 1. b) require that all software meet applicable QA requirements using a graded approach. Therefore DOE O 414.1D applies to both safety software for nuclear facilities and OSW with safety (other than nuclear), mission critical, research or general applications. Attachment 2 of DOE O 414.1D lists the ten QA criteria as requirements to be implemented for all software using a graded approach. Section 4 and Appendix A of this Guide provide a discussion of these ten QA criteria as they apply to software.

Organizations should also coordinate with the Office of the Chief Information Officer regarding corporate SQA procedures. In particular DOE O 200.1A[1] should be consulted regarding the acquisition, use and management of software to meet program and mission goals.

1.5 SOFTWARE QUALITY ASSURANCE PROGRAM

All software-related activities should be performed under a DOE-approved QAP developed by the contractor using a graded approach. The QAP should identify the applicable industry and consensus software standards used as part of the SQA work activities.

The DOE O 414.1D defines "graded approach" as:

The process of ensuring that the levels of analyses, documentation, and actions used to comply with requirements is commensurate with:

(1) The relative importance to safety, safeguards and security;

(2) The magnitude of any hazard involved;

(3) The life-cycle stage of a facility or item;

(4) The programmatic mission of a facility;

(5) The particular characteristic of a facility or item;

(6) The relative importance to radiological and nonradiological hazards; and

(7) Any other relevant factors.

Using the above criteria, the graded approach to QA for software should take into consideration the following:

- Software's intended function (e.g., safety, safeguards and security, basic facility operations);
- Software category (e.g. monitoring and control, design analysis);
- Software type (e.g., acquired, custom-developed, configurable);
- Consequences of software failure on facility's operation (e.g., life-safety, environmental releases; mission failure);
- Complexity of the software; and
- Useful life of the software (e.g., one-time use).

[1] DOE O 200.1A, Information Technology Management, dated 12-23-2008

2. SOFTWARE TYPES, CATEGORIES AND GRADING

This section provides guidance on classifying software by type and category and on grading software for SQA applications. Classification determines which SQA requirements from DOE O 414.1D apply to and guide implementation of those requirements.

2.1 SOFTWARE TYPES

Software applications are classified as acquired, configurable, or custom-developed software types. The following provides guidance to support the classification.

2.1.1 Acquired Software

Acquired software is generally supplied through basic procurements, two-party agreements, or other contractual arrangements. Acquired software includes commercial off-the-shelf (COTS) software, where (a) the software supplier does not advertise any willingness to modify the software for a specific customer and (b) fitness for use has been demonstrated by a broad spectrum of commercial users. Acquired software also includes government-furnished software downloaded at no cost to the user (referred to as “freeware”), and software acquired from a parent corporate entity or from an open source distributor.

Examples of acquired software include software used for design analysis, safety and hazard analysis, modeling of physical phenomena and engineering, inventory management, project management, corrective action tracking, and design analysis software used by design services contractors. Acquired software also includes operating systems, compilers, software development tools and firmware where software is embedded in the hardware and cannot be modified by the end user after receipt.

2.1.2 Configurable Software

Configurable software is commercially available software or firmware that allows the user to modify the structure and functioning of the software in a limited way. Examples of configurable software include programmable logic controllers (PLC), intelligent digital devices, spreadsheets, databases, and commercial calculational software such as Statistica, MathCad, Mathematica, and Matlab. Configurable software also includes utility calculation software used to create user developed algorithms, data structures, or other simple software products using commercial software.

2.1.3 Custom-Developed Software

Custom-developed software is a one-of-a-kind, unique software application built specifically for DOE or its contractors. Custom software may be developed by DOE, by an existing DOE contractor, or by a qualified software development company selected via the procurement process. Examples of custom-developed software include custom material inventory and tracking software, custom data management software, custom accident consequence analysis software, custom control system software, custom operations software, and custom research software.

2.2 SOFTWARE CATEGORIES AND GRADING

There are two software categories: (1) safety software and (2) OSW. Each category has three subcategories.

Safety software is defined by DOE O 414.1D and includes:

- Safety System Software (SSS);
- Safety and Hazard Analysis and Design Software (SHADS); and
- Safety Management and Administrative Controls Software (SMACS).

OSW includes:

- Safety-Affecting Software (SAS);
- Critical Software (CS); and
- General Software (GS).

2.2.1 Safety Software

2.2.1.1 Safety System Software

DOE O 414.1D defines SSS as:

> Software for a nuclear facility that performs a safety function as part of an SSC and is cited in either (a) a DOE-approved documented safety analysis; or, (b) an approved hazard analysis per DOE P 450.4, *Safety Management System Policy*, dated 10-15-96 (or latest version)[2] and 48 CFR 970-5223-1.

The use of the term "nuclear facility" includes both reactor and nonreactor nuclear facilities (i.e., Hazard Category[3] 1, 2, 3 and below Hazard Category 3). Safety analysis for nuclear facilities classified as Hazard Category 1, 2, and 3 are provided in a Documented Safety Analysis (DSA). Because a DSA is not developed for below Hazard Category 3 facilities (i.e., radiological facilities), the hazard analysis and hazard controls for such facilities are provided in a documented safety management system per DOE P 450.4A and the DEAR[4] clause.

[2] DOE P 450.4 is canceled and replaced by DOE P 450.4A, Integrated Safety Management Policy, dated 4-25-2011
[3] DOE-STD-1027-92, *Hazard Categorization and Accident Analysis Techniques For Compliance With DOE Order 5480.23, Nuclear Safety Analysis Reports* December 1992, (Change Notice No. 1, dated September 1997)
[4] Department of Energy Acquisition Regulation, 48 CFR 970-5223-1

When systems depend on software to implement a safety function identified in the DSA, technical safety requirements (TSR), or other formal hazard analysis, that software becomes safety software. It is possible that the software is not explicitly identified in the DSA, TSR or other hazard analysis documents, but the safety function performed by the software would be identified.

DOE-STD-3009-2014, *Preparation of Nonreactor Nuclear Facility Documented Safety Analysis*, classify safety-related structures, systems, and components (SSCs) as safety class (SC) or as safety significant (SS) as defined in 10 CFR 830.3. For SSS purposes, a "safety function" is interpreted as defined in a DSA that states the reasons for designating the SSC as SC or SS, or states the rationale for designation as a specific administrative control (SAC), and describes its preventive or mitigative safety function(s) as determined in the hazard and accident analysis. [5]

2.2.1.2 Safety and Hazard Analysis Software and Design Software

DOE O 414.1D defines SHADS as:

> Software that is used to classify, design, or analyze nuclear facilities. This software is not part of an SSC but helps to ensure the proper accident or hazards analysis of nuclear facilities or an SSC that performs a safety function.

Safety and Hazard Analysis Software:[6] This category includes software relied upon to select hazard controls and physical features that are credited to prevent or mitigate a hazard to the workers, collocated workers, the public and the environment. Software used to select or design controls for fire protection, nuclear criticality, or natural phenomena hazards for nuclear facilities are included in this safety software definition.

Design Software: [4] The software that meets this definition is relied upon to ensure that the nuclear safety design criteria are met. DOE O 420.1C, *Facility Safety*, Attachment 2, Chapter I, "Nuclear Safety Design Criteria" describes some of the analyses to be performed during design and construction. This software generally does not include computer-aided design (CAD) software, or software that supports standard facility engineering activities.

2.2.1.3 Safety Management and Administrative Controls Software

DOE O 414.1D defines SMACS as:

> Software that performs a hazard control function in support of nuclear facility or radiological safety management programs or technical safety requirements or other software

[5] Adopted in part from Software Quality Assurance Support Group (SQASG) Technical Report (TP)-07-01 Rev 2, DOE Safety Software Examples, Section 2.1
[6] Adopted in part from SQASG TP-07-01, Rev 2, DOE Safety Software Examples, Section 3.1

that performs a control function necessary to provide adequate protection from nuclear facility or radiological hazards. This software supports eliminating, limiting, or mitigating nuclear hazards to workers, the public, or the environment as addressed in 10 C.F.R. Parts 830 and 835, the DEAR Integrated Safety Management System clause, and 48 CFR 970-5223-1.

This software does not directly affect an SC/SS SSC safety function. Safety management programs and administrative controls relied upon for safety are identified in the facility DSA and TSRs. (See DOE O 420.1C and DOE-STD-3009-2014 for additional information on the types of hazard controls that would meet this definition.

These hazard control functions for non-SC/SS SSCs may be identified in the DSA for safety-related equipment in accordance with the safety management programs. The DSA hazard evaluation process may identify preventive or mitigative controls that do not rise to the level of SC or SS but still enhance the safety of the facility. These controls are identified in the hazard evaluation table, but are not explicitly credited with a SC/SS designation as identified in the DSA. Such controls are maintained in accordance with safety management programs.

Examples of SMACS include monitoring software that (a) assists operators in deciding what actions to take or not take for safe operation or (b) provides operators with visual or audio alerts of process upsets or an actual or potential unsafe condition; or (c) identification of actions, that if not taken, could cause criticality limits to be exceeded For software to be classified as SMACS, the software hazard control function should be identified in the DSA or TSR or radiological facility safety management system.

2.2.1.4 Safety Software Determination

The determination of what constitutes safety software should be made based on its application and the safety consequences of its postulated failure to perform its intended function. The process for identifying safety software is tied to the formal safety analysis and hazard assessment process for the nuclear facility. Since software can have both safety and non-safety applications, a determination should be made whether the non-safety application could adversely affect the safety application. This determination is also important where software is used to monitor radiological inventory as a means of categorizing a nuclear facility.

2.2.1.5 Safety Software Quality Assurance Grading

DOE O 414.1D requires sites to develop and implement safety software grading levels approved by DOE. Software must be appropriately categorized as safety software prior to addressing safety software grading levels.

In Table 2.2-1, three grading levels are suggested for safety software. Other grading levels may be used. The grading levels are hierarchical from most important to least important in terms of the potential consequences of failure. Safety software that does not meet the criteria for either level A or level B is automatically categorized as level C.

Table 2.2-1: Safety Software Grading Levels

Grading Level Description	Grading Levels
Safety software must execute correctly or the intended use of the system/software will not be realized, causing one or more of the following grave consequences: • Adversely affect the safety function of a SC or SS SSC cited in a DOE-approved DSA; • Compromise a limiting condition for operation or limiting control settings of a SC or SS system, as stated in a TSR; • Adversely affect the safety or hazard control function of other systems, such as toxic or chemical protection systems, cited in either (a) a DOE-approved DSA or (b) a radiological facility Safety Management System per DOE P 450.4A, and the DEAR clause (48 CFR 970.5223-1); • Result in non-conservative hazard analysis, accident analysis or hazard control selection per 10 CFR Part 830 or DOE-STD-3009-2014; or • Adversely affect the safety function of a SAC, or the supporting SSCs to implement a SAC, cited in a DOE-approved DSA.	A
Safety software must execute correctly or the intended use of the system/software will not be realized, causing one or more of the following serious consequences: • Adversely affect the hazard control function of defense-in-depth SSCs identified in a safety management program, in a DOE-approved DSA; or • Result in inaccurate or non-conservative design and analysis of SC or SS SSCs cited in a DOE-approved DSA; or • Result in an incorrect analysis, monitoring, alarming, or recording of hazardous exposures to workers or the public.	B
Safety software must execute correctly or an intended hazard control function will not be realized, causing minor consequences. This grading level will apply to all safety software not graded as level A or B.	C

2.2.2 OSW

This Guide introduces the term OSW to include software not designated as safety software. OSW includes software that is used for worker occupational safety and health, environmental compliance and protection, emergency preparedness and response, safeguards and security, mission-critical operations, project-critical activities, and research and development activities that do not impact nuclear safety. OSW has to meet the applicable QA requirements in DOE O 414.1D, Attachment 2, using a graded approach.

2.2.2.1 Safety-Affecting Software (SAS)

SAS includes software that performs a hazard control function that does not meet the nuclear safety software definitions in DOE O 414.1D. This software contributes to the control of radiological, biological, chemical, and physical hazards defined by the site's Integrated Safety Management System (ISMS) hazard analysis process. Such software may be relied upon for the control of hazardous chemicals, explosives, asphyxiants, high energy chemical reactions, combustible gases, toxic materials, nerve agents or biological agents. SAS includes software whose failure could have a high to very high (See Table 2.2-2, OSW Grading Levels) safety or health consequence and therefore should be subjected to a high degree of SQA rigor.[7]

2.2.2.2 Critical Software (CS)

The term critical software is used in this Guide for software relied upon to keep a facility or process operating as desired. Examples include software for environmental protection, emergency preparedness and response, communication, accident mitigation and recovery, regulatory compliance, hazardous inventory management, waste management, safeguards and security, machine control or machine protection systems and some research and development activities. This category involves software whose failure could pose a moderate to high consequence but not a direct health or safety consequence. This category of software should be subjected to a moderate to high (See Table 2.2-2, OSW Grading Levels) degree of SQA rigor.

2.2.2.3 General Software (GS)

The term general software is used in this Guide for software not falling into any of the software subcategories described above. Failure of this software would be of low (See Table 2.2-2, OSW Grading Levels) consequence, justifying a lower degree of SQA rigor.

2.2.2.4 OSW Determination

The determination of what constitutes OSW should be made by the organization using the software. When identifying SAS, consideration should be given to the provisions of 48 CFR 970.5223-1, *Integration of Environment, Safety, and Health into Work Planning and Execution*, and 10 CFR Part 851, *Worker Safety and Health Program*, which respectively establish requirements for the integration of environment, safety and health program principles into work planning and execution and the identification of workplace hazards and accompanying hazard analyses.

[7] Rigor, as used in this Guide, refers to the level of detail and thoroughness in the analysis, documentation and action used to satisfy a requirement.

2.2.2.5 Quality Assurance Grading of OSW

Attachment 1 of DOE O 414.1D requires that a contractor apply the general QA principles of the DOE O 414.1D, Attachment 2 to “all software”. This Guide suggests the use of four grading levels for OSW, though other approaches and levels may be used. The grading levels are hierarchical from most important to least important in terms of the potential consequences of failure. In terms of risk, the grading levels represent the relative risk associated with the failure of the software. The graded approach for OSW should provide that software with a higher risk is subjected to more rigorous SQA than software associated with moderate or low risk. The graded approach should be documented.

Table 2.2-2 on the next page provides an example of how OSW grading levels may be defined. Guidance on how SQA work activities can be graded is found in Section 5.1 of this Guide. This approach can be applied to an entire system or to individual software components.[8]

[8] IEEE Std 730-2014, IEEE Standard for Software Quality Assurance Processes, Annex I, Section 1.1.

Table 2.2-2 OSW Grading Levels

Grading Level Description	Risk	Grading Levels
OSW must execute correctly or the intended use of the system/software will not be realized, causing one or more of the following consequences: • Loss of human life or injuries resulting in long term disability; • Disabling of a hazard control of radiological (not covered by 10 CFR Part 830), biological, industrial, accelerator, laboratory, worker safety and health, or higher-level hazards credited per DOE P 450.4A, *Integrated Safety Management Policy*, and the DEAR clause; • Significant offsite release to the environmental; or • Serious impact on mission or performance objectives.	High to Very High	4
OSW must execute correctly or the intended use of the system/software will not be realized, causing one or more of the following consequences: • Injuries resulting in short term disability; • Reduction in the hazard control of radiological (not covered by 10 CFR Part 830), biological and industrial hazards, accelerator, laboratory, worker safety and health, or higher-level hazards identified by the site's hazard categorization process; • Incorrect analysis, monitoring, alarming, or recording of hazardous exposures to workers or the public which may result in non-compliance; • Significant onsite release to the environmental; • Significant regulatory violation; or • Moderate impact on mission or performance objectives.	Moderate to High	3
OSW must execute correctly or the intended use of the system/software will not be realized, causing one or more of the following consequences: • Non-reportable injury; • Release to the environment to a specific building or facility; or • Low impact on mission or performance objectives.	Low to Moderate	2
OSW must execute correctly or the intended use of the system/software will not be realized, causing minor consequences. This grading level will apply to all OSW not graded as level 4, 3 or 2.	Low	1

3. SOFTWARE MANAGEMENT

Section 3 provides guidance on several software management topics that supports SQA.

Section 3.1 provides guidance on safety software quality assurance (SSQA) including guidance on two special features of safety software required by Appendix D of DOE O 414.1D followed by information on the DOE Safety Software Central Registry and how it can be used to support SSQA at DOE and legacy software applications.

Section 3.2 provides guidance on a several topics applicable to safety and "other" software including:

- Use of National/International Consensus Standards
- Continuous Improvement, Measurement, and Metric
- Software Reliability
- System, Software and Risk

3.1 SAFETY SOFTWARE IDENTIFICATION AND MANAGEMENT

This section addresses the role of design authority in safety software identification and management of the safety software inventory.

3.1.1 Role of the Design Authority in Safety Software Identification

DOE O 414.1D, Attachment 4, requires, as applicable, involvement of the facility design authority in the identification of: requirements specification, acquisition, design, development, verification, and validation including inspection and testing, configuration management, maintenance, and retirement of safety software. How this involvement should occur is best addressed in the description of the design authority's software responsibilities. Involvement of the design authority ensures that safety application of software is properly identified and documented.

Design authority involvement may be further defined by the selected consensus standards in accordance with the QAP. For example, it may require design authority approval of such standards prior to implementation of the design basis, approval prior to performing tests[9] that require temporary changes to the approved configuration of a facility.[10]

[9] ASME NQA-1-2008/NQA-1a-2009, Part I, Requirement 3.
[10] ASME NQA-1-2008/NQA-1a-2009, Part I, Requirement 11.

3.1.2. Safety Software Inventory Management

DOE O 414.1D Attachment 4, requires sites to identify, document, control and maintain a safety software inventory. The intent of such inventory management is to provide a verifiable list of safety software which should be useful in the event of potential software issues. Table 3.1-1 shows the content of the inventory required by DOE O 414.1D. In addition to the inventory's technical content requirements noted in Table 3.1-1, document control and records management principles should be applied to software according to the provisions of the QAP.

Table 3.1-1 Safety Software Inventory Requirements Based on DOE O 414.1D

Requirement Reference and Title	Requirement Summary
DOE O 414.1D, Attachment 4, Section 2.a (2)	Identify, document, control and maintain safety software inventory. The inventory entries must include at a minimum the following: 1. software description; 2. software name; 3. version identifier; 4. safety software designation (e.g., SSS, SHADS, SMACS); 5. grade level designation; 6. specific nuclear facility application used; and 7. responsible individual.

The following guidance is provided to further aid in development and maintenance of a software inventory:

- An inventory is only as good as the information that it contains. The inventory should be developed and maintained with quality processes to ensure information integrity and user confidence.
- Various inventory formats may be used. A site inventory may be a single inventory or a compilation of various inventories.
- The level of detail for the facility application should be sufficient to provide identification and traceability to the specific facility application.
- Periodic assessments and audits of the inventory and its management system should promote inventory integrity.

3.1.3 Management of Central Registry and Toolbox Codes

A toolbox code is a term used to identify software qualified to be listed in the DOE Safety Software Central Registry (http://energy.gov/hss/services/assistance/quality-assurance/safety-software-quality-assurance-central-registry) that is used primarily for DOE safety analyses. These codes have widespread use throughout the DOE complex and are of appropriate qualification for use at DOE sites. These codes, including specific versions, have been evaluated using SQA criteria for safety software and are generally found to meet DOE SQA requirements. The analysts using a specific approved version of these codes should not have to further justify their qualification. However, other SQA requirements according to the established site procedures should be followed.

The majority of the codes in the current Central Registry have been developed outside of DOE, in the private sector or by other federal agencies). Access to the toolbox codes is subject to agreements, conditions, and restrictions established by the code custodians or federal agencies. Use of the Central Registry toolbox codes is not mandatory.

The Central Registry collection of safety software applications evolves over time. New versions of software may be added to the Central Registry, while older versions may or may not be removed. Some software may be retired and not allowed for use in DOE safety analysis. Moreover, new software may be added through the formal "toolbox-equivalency" process, having been recognized as meeting the equivalency criteria. Appendix B of this Guide addresses the process for adding new software applications and versions to, and removal of retired software from, the Central Registry.

3.1.4 Legacy Software Management

Legacy software is existing software that (a) has not been previously approved under a QA program consistent with DOE O 414.1D and (b) has been identified as safety software to be evaluated using the graded approach. In many cases, legacy software originally met DOE or industry requirements, but the SQA practices for the software were not updated as the SQA standards changed.

Legacy safety software should be identified and controlled prior to evaluation using the graded approach framework in this Guide. The evaluation should be sufficient to adequately address the correct operation of the safety software in the environment where it is being used. This evaluation should include: (1) identification of the capabilities and limitations for intended use; (2) any test plans and test cases to demonstrate those capabilities; and (3) instructions for use within the limitations.[11] One example of this evaluation is an *a posteriori* review.[12] Future modifications to existing safety software should meet all safety software work activities in DOE O 414.1D that are associated with the modifications.

For other legacy software, a graded approach may be used to ensure that the quality attributes of such software are consistent with its intended use.

[11] ASME NQA-1-2008/NQA-1a-2009, op. cit., Part II, Subpart 2.7, Section 302
[12] American National Standards Institute (ANSI)/American Nuclear Society (ANS) 10.4-1987 (R1998), *Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry,* ANS, 1998, Section 11.

3.2 SOFTWARE QUALITY ASSURANCE ELEMENTS

3.2.1 Use of National/International Consensus Standards

DOE O 414.1D requires the use of applicable national or international consensus standards to develop and implement a QAP. Use of consensus standards facilitates a common software quality engineering approach to developing or documenting software, based upon a consensus of experts in the particular topic areas.

For Hazard Category 1, 2 and 3 nuclear facilities, DOE O 414.1D cites the American Society of Mechanical Engineers (ASME) standard NQA-1-2008/NQA-1a-2009, *Quality Assurance Requirements for Nuclear Facilities*, as an acceptable QA standard for acquiring, developing and implementing safety software. Other national or international consensus standards of equivalent quality may be used for safety software. Any alternate consensus standard used should be specified in the site QAP, with DOE's approval.

For safety software used in other facilities and activities, DOE O 414.1D cites examples of appropriate standards that may be used including ASME NQA-1-2008/NQA-1a-2009 and ASME NQA-1-2000. For OSW, any of the cited standards may be used to meet the ten DOE QA criteria found in 10 CFR Part 830 Subpart A and DOE O 414.1D. DOE's criteria are mapped to the applicable ASME NQA-1 criteria in Subpart 4.5, Table 400, of ASME NQA-1. DOE O 414.1D requires that for safety software, additional standards are to be used if the selected QA standard does not adequately cover specific work activities.

Appendix D of this Guide provides references to other standards which may be useful in applying industry practices for SQA.

3.2.2 Continuous Improvement, Measurement, and Metrics

Measurements and metrics used throughout the software life-cycle should bolster confidence that the software will function in a safe and reliable manner. Design, testing, and software reliability measures are essential indicators to assess the safe and reliable performance of the software.

DOE O 414.1D, Criterion 3, *Quality Improvement*, specifies that processes should be established and implemented to detect and prevent problems. Measurements and metrics can be leading indicators for potential future problems, permitting preventive actions to be taken. For long term avoidance of problems, continuous improvement methods should be implemented. Metrics provide qualitative or quantitative indicators of improvements or lack thereof when a process or work activity has been modified. Institute of Electrical and Electronic Engineers (IEEE)

Standards 982.1[13]and 982.26[11] provide recommendations for what metrics to use and when their application is most appropriate.

3.2.3 Software Reliability

Reliability should be designed into a system, just as quality should be built into the system. The reliability of a system in which software is a subcomponent can be enhanced by the application of two primary approaches to the software design process: (1) applying good engineering practices based upon industry standards and best practices and (2) guiding design through the results of an analysis of the software failure modes.[14] Yet identifying and assessing the software failure modes is not enough to make a system reliable; the information from the failure analysis must be factored into the design.

Applying industry-accepted software engineering and software quality engineering practices is generally the first approach to developing high-quality software systems. These practices can be applied to software to improve the quality and add a level of assurance that the software will perform its safety functions as intended. Attachment 4 of DOE O 414.1D specifies the SQA work activities to be performed for safety software. However, these work activities can also be applied to OSW. Many national and international consensus standards, such as ASME NQA-1-2008 NQA-1a 2009, ANSI/ANS 10.4, and the IEEE software engineering series, provide detailed guidance for performing these work activities.

Software process capability models, such as the Software Engineering Institute's legacy Software Capability Maturity Model (SW-CMM) and its more recent Capability Maturity Model Integration (CMMI)[15] are proven SQA tools. The CMMI offers two approaches, staged and continuous. Organizations introducing a software process improvement program should consider using these models.

3.2.4 System, Software and Risk

Software is typically a component of a system much in the same way hardware, data, and procedures are system components and rarely function as an independent entity. Therefore, to understand the risk associated with the use of software, the software function should be considered a part of an overall system function. Managing the risk appropriately is the key to managing a software system.

---

[13] IEEE Std 982.1-1998, *IEEE Standard Dictionary Of Measures To Produce Reliable Software*, IEEE, 1998.
[14] Identifying Software Failures Using FMEA and Fault Tree Analysis, *Quality Progress*, ASQ, September 2012
[15] Capability Maturity Model Integration (CMMI) Product Team, Capability Maturity Model Integration, Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), Continuous Representation, CMU/SEI-2002-TR-011, ESC-TR-2002-011, Carnegie Mellon University (CMU) Software Engineering Institute (SEI), 2002.

The risks and consequences of software failures should be addressed in terms of the contribution of a software failure to an overall system failure. Issues such as safeguards and security, training of operational personnel, electromagnetic interference, human-factors, or system reliability have the potential to contribute to the failure of a system. For example, if the security of the system can be compromised, then the associated software within the system can also be compromised. Controlling access to the system is vital to maintaining the integrity of the software, and maintaining the confidentiality of the data that the software processes or produces. In another example, considering PLCs or network software applications, electromagnetic interference could offer potential risks for operation of the software system.

Once the software's function within the overall system's function is established, and the consequences of the potential failures are identified, the appropriate software life-cycle work practices can be identified to minimize the risk of software failure on the system. Rigor can then be applied commensurate with the risk to be managed.

SQA cannot address the risks created by the failure of other system components (i.e., hardware, data, human process errors, power system failures) but it can address the software "reaction" to effects caused by these types of failures. SQA should not be isolated from system-level QA and other system level activities. In many instances, hardware fail-safe methods are implemented to mitigate risk of software failure as part of the SQA process. Additionally other interfaces such as hardware and human interfaces with software should implement QA activities.

4. MANAGEMENT OF SOFTWARE BY TEN QA CRITERIA[16]

DOE O 414.1D, Attachment 2, Quality Assurance Criteria, lists the ten QA criteria to be applied to all software (safety software and OSW) using a graded approach. Therefore, a DOE contractor's QAP should address how each of the ten QA criteria should be applied to software. The application of each of the ten QA criteria to discussed in Appendix A.

For safety software, DOE O 414.1D, Attachment 4 lists ten work activities that are to be implemented, as applicable, for safety software quality assurance (SSQA) using the consensus standard selected and grading levels established. The SSQA work activities are derived from common industry practices and can be applied using the graded approach to all software. A discussion of how the ten SSQA work activities could be applied to generally satisfy the ten QA criteria for software acquisition, development, and implementation is provided in Appendix A.

[16] This section on application of the ten QA criteria replaces a section on "Recommended Practices" in the previous version of this Guide.

5 SELECTING AND IMPLEMENTING SOFTWARE QUALITY ASSURANCE WORK ACTIVITIES

5.1 OVERVIEW OF SOFTWARE-RELATED WORK ACTIVITIES

This section provides an overview of how to implement the applicable SQA work activities required by DOE O 414.1D. Attachment 4 of DOE O 414.1D requires contractors to use the selected consensus standards and grading levels established and approved by DOE to implement these SQA work activities:

(a) Software project management and quality planning;
(b) Software risk management;
(c) Software configuration management;
(d) Procurement and supplier management;
(e) Software requirements identification and management;
(f) Software design and implementation;
(g) Software safety analysis and safety design methods;
(h) Software verification and validation;
(i) Problem reporting and corrective action; and
(j) Training of personnel in the design, development, use, and evaluation of safety software.

A step-by-step approach is presented that may be used for selecting and implementing SQA work activities. Although these work activities are specifically indicated in DOE O 414.1D to be applicable for safety software, they also represent standard SQA activities generally followed in the software industry. The approach may therefore be used for both safety software and OSW. Other approaches may also be acceptable as long as they meet the DOE QA requirements. Additional information, including several examples, is provided in Appendix C to further illustrate the approach. Section 5.2 of this Guide provides additional guidance specific to each of the ten SQA work activities.

As shown in Figure 5.1-1, SQA work activities follow a six-step process:

1. Identify applicable national or international consensus standards,
2. Determine the software category, subcategory, and grading level,
3. Determine applicability of the work activity,
4. Define the work activity using applicable consensus standard,
5. Grade by applying appropriate work activity requirements, and
6. Grade by applying appropriate rigor.

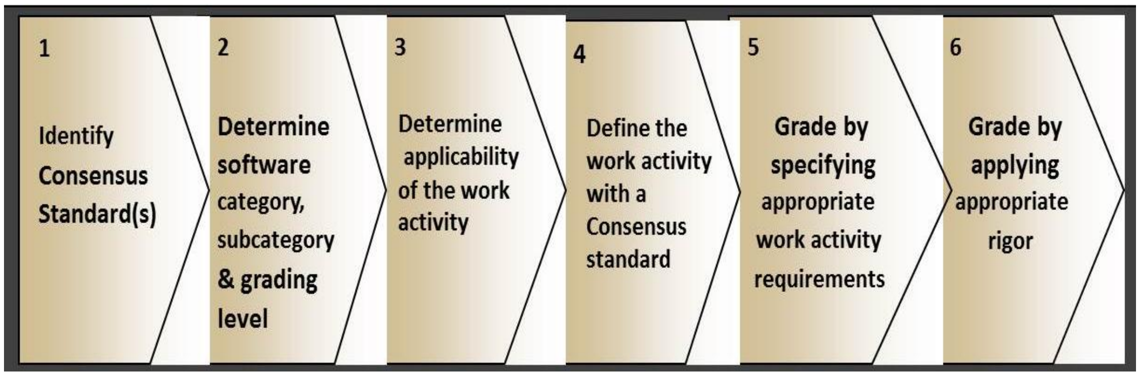
1
Identify Consensus Standard(s)
2
Determine software category, subcategory & grading level
3
Determine applicability of the work activity
4
Define the work activity with a Consensus standard
5
Grade by specifying appropriate work activity requirements
6
Grade by applying appropriate rigor

Figure 5.1-1 A Six-Step Approach to Selecting and Implementing Work Activities

The first step, identifying consensus standards, as required by DOE O 414.1D should be addressed in the site's QAP.

The second step is to determine the software category (i.e., safety or OSW), subcategory and grading level.

The third step is to determine the applicability of the work activity. Identify and apply work activities if they are: (a) within the scope of the software project, that is germane to the software/work at hand, and (b) within the control of the project. If the work activities are not within the scope or control of the software project, they are not applicable.

The fourth step is to define the work activities with the appropriate elements from the selected consensus standards. This is done by specifying which elements, in whole or in part, of the selected consensus standards describes the work activity.

The fifth step is to define appropriate requirements of the elements to be used for each work activity based on the software category and grading level.

The final step is to grade by applying appropriate rigor.

For additional detail and examples regarding the steps in selecting and implementing applicable work activities, see Appendix C.

5.2 DETAILED GUIDANCE ON SOFTWARE-RELATED WORK ACTIVITIES

The SQA work activities described in this section, which include tasks that are specific to software life-cycle phases, provide the basis for planning, implementing, maintaining, and operating software. Sections 5.2.1 through 5.2.10 describe the elements of these work activities. Note that:

- These work activities describe the standard life-cycle process generally followed in software development;
- Not all of these work activities are applicable for all types software;
- These work activities are applied in a graded manner depending on the software categories/subcategories and grading level;[17]
- Greater rigor is applied throughout where safety is involved; and
- These work activities are configured so that they address all ten QA criteria in Appendix A.

5.2.1 Software Project Management and Quality Planning

5.2.1.1 Safety Software Project Management and Quality Planning

Software, project management starts at the system level. Software-specific tasks should be identified and included either within the overall system planning or in separate software planning documents. These tasks may be documented in a software project management plan (SPMP), a software quality assurance plan (SQAP), a software development plan (SDP), or similar documents. They also may be embedded in the overall system-level planning documents. Typically, the SPMP, SQAP, or SDP are the controlling documents that define and guide the processes necessary to satisfy project requirements, including the software quality requirements. These plans are initiated at the beginning of the project life-cycle and are maintained throughout the life of the project, through retirement of the software. Retention beyond this point is governed by applicable record retention requirements.

Software planning templates and associated planning guidance documents and tools are available in nationally-recognized standards.[18] These proven standards and other equivalent references should be used whenever possible for project planning. The project management templates and guidance add important project management elements, such as cost and schedule, to the technical elements of DOE O 414.1D and the selected consensus standard and hence help to ensure the successful completion of the software project.

Work activity requirements, grading and rigor should be described in the project planning documentation. The following general guidelines, based in part on lessons learned from previous DOE projects should be considered in software project planning:

- Planning documentation should be kept current and updated as the project moves forward. At the onset of a software project, the planning documentation may focus on

[17] SQASG-TP-14-01 REV. 0, *DOE Nuclear Safety Software Graded Approach Assessment Tool*, Appendix C provides grading examples based on software work activities.
[18] IEEE/ International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 16326-2009, *Systems and Software Engineering--Life Cycle Processes--Project Management*

how to develop and design the software, with less emphasis on the details of software use. As the software is developed and released, the project planning documentation should be updated to focus on use, maintenance, inventorying, and retirement of the software.

- Planning documentation should be developed to a level of detail that will promote successful execution of the project.
- Planning documentation should be integrated with DOE cyber-security planning requirements and documentation whenever possible to ensure integrated software items and services.
- Planning documentation should address and integrate all applicable elements of the DOE O 414.1D, including suspect/counterfeit item prevention and software inventory.
- Planning documentation should clearly identify SQA requirements and grading, and provide a supporting rationale. This includes providing a clear description of the applicability of the requirements, the grading by specifying the appropriate work activity requirements, and the grading by applying appropriate rigor.
- Planning documentation should delimit the scope of the project and show that the scope of software work can be managed by the project team.
- Planning documentation should address the financial and human resources available, project schedule constraints, and methods for controlling changes to both resources and schedule.
- Planning documentation should be developed for all software including custom-developed, configurable or acquired.

Key aspects of software project planning include project management, acquisition, design, verification and validation, approval for use, inventory, use and maintenance, and retirement. [19] Key aspects of software project management include the application and documentation of standards and conventions; categorization and grading; roles and responsibilities for the software project team; interface control; problem detecting, reporting, preventing/mitigating and corrective action; and configuration management of documentation, computer programs, and support software. Some of these project planning elements may be addressed in other life-cycle documentation.

5.2.1.2 Project Management and Quality Planning for OSW

Project management and quality planning apply to all software based on the graded approach. However, grading is more commonly applied to OSW than to safety software because failure consequences associated with OSW are lower.

[19] Records planning/retirement should consider the need for records retrievability and future use of software and as applicable, support software and hardware

To grade OSW, the safety software guidance in the previous section may be used. Grading is done by specifying the appropriate work activity requirements and by applying the appropriate rigor to implementation Appendix C offers an example that illustrates these points OSW. Although this example applies to OSW, its general concepts may be applied to safety software.

5.2.2 Software Risk Management

There are increased project risks associated with the successful completion of a software project with respect to custom-developed and configurable software. Software project risk management provides a disciplined process for assessing the risk for potential software project failure. Successful completion of a software project depends on careful identification of failure risks and what actions can be taken to minimize those risks. [20,21, 22] Risk associated with the failure of a software to perform its intended function is addressed separately in Section 5.2.7, *Software Safety Analysis and Safety Design Methods* .

Managing project risks comprises three activities, namely risk assessment, risk control, and resolution.

Risk assessment means identifying potential risks, analyzing them, and prioritizing them to ensure that the available resources will be efficiently used. Several risk identification techniques are described in literature and consensus standards and literature. [23,24]

Risk control means tracking and resolution of risks. Tracking efforts should cover operational and development costs, availability of resources, schedule constraints, and technical aspects of the project.

Risk resolution means avoidance or mitigation of risk. Risk resolution is important because even small risks during one phase of the software life-cycle have the potential to increase in some of the downstream phase with possible very adverse consequences.

The SPMP (or a separate risk management plan) should specify how project risks will be managed. The following elements should be covered:

- procedures for contingency planning;

[20] SQAS21.01.00-1999 (Reference Document), *Software Risk Management: A Practical Guide*, Department of Energy Quality Managers Software Quality Assurance Subcommittee, dated 2-2000.
[21] IEEE/ISO/IEC 16326-2009, *Systems and Software Engineering--Life Cycle Processes--Project Management*
[22] ISO/IEC 16085 IEEE Std 16085, *Systems and Software Engineering-Lifecycle Processes-Risk Management.*
[23] Christensen, Mark J., and Richard H. Thayer, *The Project Manager's Guide to Software Engineering's Best Practices***,** Institute of Electrical and Electronics Engineers Computer Society Press, 2001, pp. 417–4.
[24] Society of Automotive Engineers (SAE) JA1003, *Software Reliability Program Implementation Guide*, SAE 2004, Appendix C4.6

- methods for tracking the various risks, evaluating changes in the levels of risk and responding to those changes;
- plans for assessing initial risk factors;
- mitigation of risks throughout the life-cycle;
- description of risk management work activities, procedures and schedules;
- documentation and reporting requirements;
- organizations and personnel responsible for performing specific activities; and
- procedures for communicating risks and risk status to acquirers, suppliers, and subcontractors.

Software project risks are risks associated the acquirer-supplier relationship; contract, technology, size and complexity of the product; development and target environments; personnel acquisition; skill levels and retention; schedule and budget; and acceptance of the product. Examples of potential software project risks include:

- incomplete or unstable software requirements;
- specification of incorrect or overly-simplified algorithms;
- hardware constraints limiting the design;
- design based upon unrealistic or optimistic assumptions;
- frequent design changes during coding;
- incomplete and undefined project and organizational interfaces;
- usage of unproven computer and software technologies;
- insufficient time for development and testing;
- undefined or inadequate test acceptance criteria; and
- quality concerns with subcontractors or suppliers.

5.2.2.1 *Graded Approach to Software Risk Management*

The grading of risk associated with software projects should vary depending on the software categories, subcategories and risk associated with the software failure.

When there is significant risk that safety software, safety-affecting software, or critical software may fail, sufficient rigor in the risk management process should be applied to ensure successful project completion.

For general software, the QAP's requirements for routine work activities are likely sufficient. For example, existing processes for acquiring non-software items such as computer hardware and office supplies may be sufficient for controlling the acquisition of general purpose software.

5.2.3 Software Configuration Management

Sound configuration management applies to all software acquisition and development activities. Software configuration management (SCM) is the process of identifying software components, controlling changes, and maintaining the integrity and traceability of the configuration. SCM applies to all types of software and to the entire software life-cycle up to retirement of the software. It provides the procedures necessary to ensure that (a) software modifications are evaluated before changes are made, (b) various software units such as software modules and libraries are examined for consistency and correctness after changes are made, and (c) software is tested and accepted according to established standards.[25]

SCM activities are designed to identify all functions and tasks necessary to manage the configuration of the software system, including software engineering items, establishing the configuration baselines to be controlled, and the software configuration change control process.[26] Each of the following five functions of SCM [27] should be addressed when performing configuration management: (1) configuration identification, (2) configuration control, (3) configuration status accounting, (4) configuration evaluations and reviews, and (5) release management and delivery.

Configuration control methods for each version or update of software should be documented. To this end, a detailed SCM plan should be written to identify the required SCM activities and the procedures that should be followed. The SCM plan should cover management, activities, schedule, resources and plan maintenance.

The SCM plan should also discuss the configuration management roles of the software project team. Topics to be covered include the roles and responsibilities of software developers, testing and acceptance personnel, documentation writers, the code custodian, and the project manager. All members of the team should be responsible for implementing the required configuration management procedures and practices in their respective areas of responsibility.

A baseline labeling system should be implemented that uniquely identifies each configuration item, identifies changes to configuration items by revision, and provides the ability to uniquely identify each configuration. This baseline labeling system is used throughout the life of software development and operation. Approved changes developed subsequent to the baseline should be added to the baseline.

---

[25] ANSI/ANS-10.7-2013 *Non-Real-Time, High-Integrity Software for the Nuclear Industry - Developer Requirements,* Section 16, "Software Configuration Management"
[26] ASME NQA-1-2008/NQA-1a-2009, op. cit., Part II, Subpart 2.7, Section 203
[27] These functions are set out in IEEE Std 828-2005, *IEEE Standard for Software Configuration Management Plans,* IEEE, 2005, Section 3.

Proposed changes to the software should be documented, evaluated, and approved for release. Only approved changes should be made to the baseline software. Software verification and validation (V&V) activities should be performed for all changes to ensure correct implementation. V&V should also include any changes to the software's documentation.

Evaluations or reviews should be conducted to verify that the software product is consistent with the configuration item descriptions and that the software and its documentation are complete. Physical configuration audits and functional configuration audits are examples of audits or reviews that should be performed.[28] SCM work activities should be applied as early in the software life-cycle as possible, beginning at the point of control of the software.

During software operations, authorized user lists can be implemented to ensure that the software use is limited to trained and authorized personnel. Authorized user lists are access control specifications that are addressed in Section 5.2.5, *Software Requirements Identification and Management*. These lists should be frequently updated to account for personnel changes.

5.2.3.1 *Grading Software Configuration Management*

For safety software and safety-affecting software, all SCM functions should be fully implemented. For critical software and general software, a graded approach may be used. The scope content and rigor with which the plan is developed, implemented and monitored will vary with the significance of a software failure on an organization's safety program or mission.

5.2.4 Procurement and Supplier Management

Procurement activities can range from the procurement of COTS software, to the purchase of compilers, or other development tools used for in-house software development, to the purchase of custom-developed software. The purpose of software procurement and supplier management activities is to ensure that the software supplier is qualified and capable of providing the products or services specified in the procurement contract.

Software procurement and supplier management should include the following activities:

- Review of the software supplier's SQA practices;
- Review of the software supplier's history of providing identical or similar products or services;
- Review of shared supplier quality information through channels such as the DOE Consolidated Audit Program and the DOE Laboratory Accreditation Program);
- Review of certifications or registrations awarded by nationally/internationally accredited third parties;

[28] Institute of Electrical and Electronics Engineers (IEEE) 1042-1987, *IEEE Guide to Software Configuration Management,* IEEE, 1987, Section 3.3.4.

- Review of SQA documents provided by the supplier; and
- Review of supplier SQA information obtained though audits, surveillances, or surveys.

Software procurement documents should include the following elements:

- Safety, security, functional, and performance requirements;
- Consensus standards to be followed by the supplier (e.g., ASME-NQA-1-2008/NQA-1a - 2009, ISO 9001-2008, *Quality Management Systems - Requirements*);
- Software development and V&V processes to be used by the supplier in clearly specified environments;
- Software documentation and training programs to be provided by the supplier;
- Criteria to be used for accepting the software product or service;
- Methods for the supplier's reporting of defects, new releases, or other issues;
- Methods for the software users to report defects and request assistance when operating the software;
- Supplier/acquirer interfaces including acquirer approvals; and
- A certificate of conformance or other self-declaration that the software meets its intended quality.

5.2.4.1 Grading Software and Procurement and Supplier Management

DOE Order 414.1D, Attachment 4 requires safety software to be acquired, developed and implemented using ASME NQA-1-2008/NQA-1a-2009, or other national or international consensus standards such as ISO 9001 that provide an equivalent level of QA. If such standards cannot be applied, commercial grade dedication should be used for safety software as set forth in the site's QAP.

Purchasers of safety-affecting or critical software should ensure that suppliers have a QAP based upon a QA consensus standard listed in DOE O 414.1D, or another appropriate standard. Purchasers of general software should ensure that the software is consistent with the requirements of DOE O 414.1D as set forth in the site's QAP.

5.2.5 Software Requirements Identification and Management

Identifying and managing software requirements provides a foundation or baseline for the software development. The requirements should be developed using a process that includes eliciting, specifying, and validating.

Elicitation is aimed at identifying software stakeholders and their needs. Requirements specification consolidates and resolves conflicts in stakeholder needs and translates those needs into clear concise requirement statements that can be met with measurable acceptance criteria.

Validation consists of a review of the requirements to ensure that they are achievable, concise, complete, and satisfying to stakeholders.

Since most software applications interact with hardware, data, and other software applications as part of system, software requirements should be derived from system requirements. This is true even if the system consists of only the processor upon which the software is executed.[29] These system requirements should be translated into requirements specific for the software.

5.2.5.1 Software Requirements Specification (SRS)

The SRS should be documented in system-level requirements documents, software requirements specifications, and procurement and other agreements as appropriate prior to software design or development activities, in order to define the scope. Format and level of detail may vary with the software type. For example, software requirements for custom-developed software will differ from requirements for configurable software.[30] The SRS should contain requirements covering hardware, functions, interfaces, operations, performance, logical databases, systems and communications, security, backup and recovery, and support. The following elements should be included in the SRS:

- Functions the software is to perform in specified environments;
- Time-related performance issues of software operation such as speed, recovery time, and response time;
- Design constraints imposed on implementation-phase activities;
- Attributes of software operation such as portability, acceptance criteria, access control, and maintainability;
- Security requirements commensurate with the risk from unauthorized access or use; and
- External interfaces and interactions with people, hardware, and other software.

Requirements identified after the initial list is compiled should be reviewed and approved and then integrated into the existing SRS.

These additional recommendations should be considered:

- SRS documents should be organized to allow traceability to both the software requirements and the software design.
- SRS documents should be reviewed to ensure they are complete, correct, consistent, clear, and verifiable.

[29] IEEE Std 730-2014, IEEE Standard for Software Quality Assurance Processes, Section 4.6.3.
[30] DOE Subcommittee on Consensus Assessment and Protective Actions Special Interest Group *SCAPA SQA Guidance – July 30, 2010.*

- SRS documents should be managed throughout the software life-cycle[31] to minimize conflicting requirements and maintain accuracy for later validation activities.
- SRS documents should be reviewed at the completion of the software development cycle to ensure that assumptions made during the software development process are consistent with the intended use of the software.

Note also that custom-developed software most likely will contain a larger number of software requirements than configurable or acquired software and thus, more formal documentation may be necessary.

5.2.5.2 Grading Software Requirements Identification and Management

SRS documents for safety software application should identify the software safety functions, the consensus standards and conventions to be applied to the software, and the documents and the records to be managed in accordance with the ASME NQA-1-2008/NQA-1a-2009 standard adopted in the site QAP.

SRS documents for safety-affecting software should identify the hazard control functions of the software and consequences associated with the software not performing as intended. If the software failure results in the very high-risk category (i.e., grading level 4), the SRS should address all the elements described in Section 5.2.5.1 using consensus standards in the site's QAP.[32] For safety-affecting software with lower consequence of failure, a graded approach should be used.

SRS documents for critical software should identify the software functions and the consequences of their failure. The SRS elements described in Section 5.2.5.1 should be graded according to risk using consensus standards in the site's QAP.

Software requirements identification and management for general-purpose software should be performed according to the site's QAP.

5.2.6 Software Design and Implementation

During software design and implementation, the software is developed, documented, reviewed, and controlled. This section elaborates on the following guidance:

- The software design elements should identify the operating system, function, interfaces, performance requirements, installation requirements, design inputs, and design constraints.

[31] IEEE-Std-830-1998, *Software Requirements Specifications*, Section 4.3
[32] IEEE-Std 1228-1994, Software Safety Plans, Sections 4.3.5, 4.3.6, and 4.3.8

- The software design should be complete and sufficient to meet the software requirements.
- The design activities and documentation should be adequate to fully describe how the software will interface with other system components and how the software will function internally.
- Data structure requirements and layouts should be constructed to fully understand the internal operations of the software.

5.2.6.1 Software Design

The objective of the design phase is to produce a detailed description of the computer program design that demonstrates how software requirements are met and provides the basis for the software development activity. In this phase, the software architecture is developed based on the software requirements. The output of the software design process is contained in one more software design description (SDD) documents. The SDD documents describe the overall software structure and its major components, and interfaces. The software design should consider both the operating and security environment and can be translated into code.

The software design should have the following features:[33]

- The description of the problem to be solved, the assumptions and limitations, mathematical models, and numerical methods to be used, should be documented
- The software should be divided into simple, manageable, and understandable sets or units (modular design),
- The critical date structures and related operations in the software design should be encapsulated within units to the maximum extent practicable to ensure that data are properly handled and no errors are encountered when the data are retrieved,
- The design should specify the interfaces (including the user interface), control flow and data flow, control logic, data structure and process structure,
- The design should implement appropriate safety design methods based on safety analysis and consider principles of simplicity, decoupling, and isolation to eliminate hazards.[34] (Complexity of the software design, including the logic and number of data inputs tends to increase the defect density in software components.),
- The design should explicitly identify data that require special handling or security protection; for access to such special data, the design should address both cybersecurity and physical security, and

[33] For additional guidance see IEEE Std 1016, *Recommended Practice for Software Design Descriptions*, 1987.
[34] Leveson, Nancy, *Safeware: System Safety and Computers*, Addison Wesley, 1995.

- The design should address and resolve potential problems that including external and internal abnormal conditions and events.

The primary deliverable in the design phase is the SDD document which should be structured and written such that a competent programmer other than the designer can implement it. The design should be documented with sufficient detail so that a qualified person can review, understand and verify its adequacy without calling upon the originator.

A good design is characterized by:

- clear traceability from the software requirements to the design,
- ease of comparing requirement specifications to overall system requirements,
- ease of comparing individual design documents to corresponding requirements specifications,
- ease of comparing program details during testing to applicable design publications, and
- ease of assessing the accuracy of implementation during the operational life-cycle phase.

For system software, the development process should be managed as an integral part of system design. Information containing system, hardware, pre-developed software requirements, and design specifications should be collected and identified as design input documentation. For additional details on digital systems software see Section 6.2.

Documentation for design and analysis software includes design and development documents and application documents. Typical design and development documentation are: problem definition, development plan, theory manual, requirements specification, design description, programmers manual, and V&V reports with version tracking. Typical application documents are: computer program abstract, theory manual, user's manual, and validation report, all with version tracking. For development, validation and evaluation of models for use with design and analysis software see Section 6.1.

Design reviews are a common tool used to ensure that the design meets specified requirements and conform to consensus standards. Such reviews are done by comparing the software product against specified criteria. Through these reviews, which should occur throughout the software's life-cycle, problems and non-conformances are identified and subsequently resolved.

5.2.6.2 Software Implementation

During implementation, the software design is translated into computer programs or code. The resulting code is debugged and integrated into computer program modules. Some of the key attributes of an effective computer program are: (1) it implements the design completely as documented in the SDD and is easily related to it; (2) it conforms to applicable programming standards; (3) it uses standard programming language with well-defined syntax and semantics;

(4) it provides protection against detectable run-time errors; (5) it has a clear and easy-to-follow program structure; (6) it facilitates verification activities and future modifications; and (7) it is easy to understand.[35]

The source code for developed software should be placed under configuration management prior to the commencement of the test phase. This test phase is often iterative in the sense that unit and integration testing are performed after code construction, the results analyzed, and then used in another round of programming to identify and correct errors.

The initial software version should be assigned prior to the completion of each software module. Changes to the software from the initial version should be controlled and documented. Documentation should include a copy of the software, test cases, and associated criteria that are traceable to the software requirements and design documentation.

Deliverables in this implementation phase include program listings, source code, executable code, and user's guide.

During implementation, static analysis, clean room inspections, and reviews are common techniques to ensure the implementation remains consistent with the design and does not add complexity or functions which could decrease the safe operation of the software. Many tools exist to evaluate the complexity and other attributes of the source code design structure. Walkthroughs and more formal structured processes for determining defects, such as Fagan inspections, can be used to identify defects in source code, design descriptions and other development process outputs.

The software developer should perform unit testing prior to system-level V&V, including acceptance testing. Developer testing can be structured and formal, using automated tools, or less formal methods. Other testing methods such as functional, structural, performance interface, stress and human-factors testing are useful to test various aspects of the software. These methods are applied to identify errors and defects and to check that the software performs as intended when errors and defects are corrected. Several tools are also available to support testing activities.

5.2.6.3 Grading Software Design and Implementation

This work activity does not apply to acquired or commercial design and analysis safety software and OSW since the design and implementation activities for this type of software.

Safety Software:

[35] IAEA Technical Reports Series No. 397, *Quality Assurance for Software Important to Safety,* IAEA, 2000.

The software design and implementation work activity as described in sections 5.2.6.1 and 5.2.6.2 above should be performed and documented for custom-developed safety software. For configurable safety software this work may be graded. The design and coding related work activities should be performed in a similar manner as with custom-developed software. Less formal design and coding reviews such as simple desk checks by another qualified individual other than the developer may be performed. Developer testing that includes safety functions, security and performance should be conducted and documented.

OSW:

The software design and implementation work activity as described in sections 5.2.6.1 and 5.2.6.2 above is applicable for custom-developed safety-affecting software with high to very high consequences of software failure. For critical and general OSW, this work activity may be graded with less formal reviews. However, the design and coding activities should preferably be performed as described in sections 5.2.6.1 and 5.2.6.2 above.

5.2.7 Software Safety Analysis and Safety Design Methods

5.2.7.1 Software Safety Analysis

The development of software applications requires (a) identification of hazards that have the potential for defeating or limiting a safety function and (b) implementation of design strategies to eliminate or mitigate those hazards. Accordingly, the software safety process should address the mitigation strategies for the components that have potential safety consequences if a fault occurs. The software design and implementation process should address the architecture of the safety software application.

Software is only one component of the overall safety system. It may be embedded in an instrumentation and control system, it may be a custom control system for hardware components, or it may be standalone software used in safety management or decision-making. In any applications of software important to safety, analysis of the application occurs first at the system level. The analysis should then be performed at the software component level to ensure adequate safeguards are provided to eliminate or mitigate the potential occurrence of a software defect that could result in a system failure. Software safety analysis is intended to identify possible hazards presented by the use and operation of the software and to describe methods that can be used to eliminate, mitigate, monitor or manage potential hazards. Typically, a hazard (i.e., abnormal conditions and events) presented by software is the result of a failure of the software to function as expected.

Software failure may have a direct or indirect effect on safety. A direct effect is one resulting in the failure, degradation or abnormal operating condition of a safety system. An indirect effect is one resulting in a reduction of the safety margin in a safety system.

Several hazard analysis techniques may be used for this purpose. These techniques include failure modes and effects analysis, fault-tree modeling, event-tree modeling, cause-consequence diagrams, hazard and operability analysis, and interface analysis. These techniques should be applied and documented when assessing the effects of credible software failures. The software safety analysis process should also address mitigation strategies for credible software failures that have safety consequences.

5.2.7.2 Safety Design Methods

Methods to mitigate the consequences of software failures should be an integral part of the software design.[36] Specific software analysis and design methods, for ensuring that safety functions are deliberately considered and properly addressed, should be performed throughout the software development and operations life-cycles. These methods include dynamic and static analyses.

During the initial conceptual design of software, potential failures should be identified and evaluated for probability of occurrence and failure consequences. Some potential problems to be watched for are: (1) complex or faulty algorithms, (2) improper handling of correct data or error conditions, (3) buffer overflow, and (4) incorrect sequence of operations due to logic or timing faults.

Hazard analysis techniques can be used for preliminary probability and consequence analyses. These analyses can later be updated as more information is known about the requirements and design structure. These techniques help practitioners understand and assess the impact of software failures on the system.

In the design architecture, safety modules should be separate from non-safety modules to minimize the effect of failure of one module on another.[37] The interfaces between the modules should be defined and tested thoroughly. Separation of the safety features also allows for more rigorous software development and verification practices to be applied to the safety components, while applying a cost-effective level of SQA to the non-safety components. Design practices should include process flow analysis, data flow analysis, path analysis, interface analysis, and interrupt analysis.

[37] IEEE Std 7-4.3.2-2003, *Criteria for Digital Computers in Safety Systems for Nuclear Power Generating Stations* Section 5.6.

When a potential hazard affecting software functions cannot be eliminated, the hazard itself should be mitigated. Given that software can experience partial failures that may degrade to an undetectable degree the capabilities of the overall system, other design techniques such as built-in fault detection and self-diagnostics should be employed. External monitors, n-version programming, and Petri nets are examples of techniques by which fault-tolerant concepts can be added to the design.[38,39] Self-diagnostics detect and report software faults and failures in a timely manner and allow actions to be taken to avoid an impact on the system's safe operation. Some of these techniques employ memory functionality and integrity tests such as checksums and watch dog timers.[40] Software control functions can be performed incrementally rather than in a single step, reducing the potential that a single failure of a software component would cause an unsafe system state.

5.2.7.3 *Grading Software Safety Analysis and Safety Design Methods*

The safety analysis and safety design methods just described apply to custom-developed and configurable software. The work activities for these methods should be carried out as indicated for Level A safety software whose failure could cause unacceptable consequences. For Level B or Level C safety software applications, the work activity may be graded. This grading may include performing a limited safety analysis to confirm that major safety risks have been identified and addressed at the software requirements and design phase. For safety-affecting software with high to very high consequences from software failure, this work activity should be conducted such that safety risks are identified and addressed.

5.2.8 Software Verification and Validation

V&V is one of the most important work activities for identifying defects, making necessary corrections, and ensuring that the corrected software will perform its intended function. DOE O 414.1D adopts the definitions for verification and validation from IEEE Std 1012,[41] as follows:

> Verification is the process of: (a) evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase; or, (b) providing objective evidence that the software and its associated products conforms to requirements (e.g., for correctness, completeness, consistency, accuracy) for all life-cycle activities during each life-cycle process (acquisition, supply, development, operation, and maintenance); satisfies standards, practices, and conventions during life-cycle

[38] Sparkman, Debra, *Techniques, Processes, and Measures for Software Safety and Reliability,* Lawrence Livermore National Laboratory, UCRL-ID 108725
[39] SAE JA1003, *Software Reliability Program Implementation Guide, SAE, 2004*, Appendix C
[40] IEEE Std 7-4.3.2-2003, op. cit., Section 5.5.3
[41] IEEE Std 1012-2004, IEEE *Standard for Software Verification and Validation*, IEEE 2004.

processes; and, successfully completes each life-cycle activity and satisfies all the criteria for initiating succeeding life-cycle activities (e.g., building the software correctly).

Validation is the process of: (a) evaluating a system or component during, or at the end of the development process to determine whether it satisfies specified requirements; or, (b) providing evidence that the software, and its associated products, satisfies system requirements allocated to software at the end of each life-cycle activity, solves the right problem (e.g., correctly models physical laws, implements business rules, uses the proper system assumptions), and satisfies the intended use and user needs.

Verification activities are performed throughout the software life-cycle, while validation activities are mainly performed at the end of software development or acquisition to ensure the software meets prescribed requirements. V&V activities should be performed by competent staff independent from those who developed the software.[42] The V&V tasks outlined in this section are derived from acceptance testing requirements in ASME/NQA-1-2008/NQA-1a-2009 supplemented by other activities, reviews, inspections, assessments, and observations described in other consensus standards.

Reviews and inspections should be performed of software deliverables[43] and processes that guide the software development activities. Traceability of the software requirements to the software design should also be performed.[44] Inspections can be formally implemented Fagan inspections, walkthroughs, or desk checks. Verification of the software design using any one of the above methods should be completed prior to approval of the software for application and use.[45] This verification process may be performed as part of the software development and implementation activity.

Observations should be performed during the development, factory or site acceptance testing, installation, and operation (i.e., in-use testing)[46] of the software. Software testing activities should be planned and implementation documented. Test cases and procedures, including expected results, should be developed and test activity deliverables such as test results and software limitations should be placed under a configuration management system.[47]

Acceptance testing should include the following tests: (1) functional testing, (2) performance testing, (3) security testing, (4) stress testing, and (5) load testing. Hazard analysis should be used for defining test cases and procedures. Testing strategies that may be appropriate for acceptance testing include: (1) equivalence class testing, (2) branch and path testing, (3)

[42] ASME NQA-1-2008/NQA-1a-2009, op. cit., Part I, Requirement 3, Section 801.4
[43] Deliverables are requirement specifications, procurement documents, software design, code modules, test results, training materials, and user documentation.
[44] ASME NQA-1-2000/NQA-1a-2009, op. cit., Part II, Subpart 2.7, Section 402.1
[45] ASME NQA-1-2008/NQA-1a-2009, op. cit., Part II, Subpart 2.7, Section 402.1
[46] ASME NQA-1-2008/NQA-1a-2009, op. cit.,Part I, Requirement 11, Section 400
[47] ASME NQA-1-2008/NQA-1a-2009, op. cit., Part I, Requirement 11, Section 200

statistical-based testing and (4) boundary value testing. User's guide, use cases, and operational profiles are instrumental in identifying and detailing the test cases and procedures.

Software used for operational control should be monitored by periodic testing throughout its lifetime to verify its current reliability and detect any degradation.[48] If such testing is not possible, monitoring using quantitative measurements should be performed.

When a new version of a software product is obtained, tests should be performed to validate that the system meets the intended requirements and does not perform any undesired functions.[49] If the system is already operational, only positive testing may be possible. In those instances analysis should be performed of failure modes to understand the consequences of failures or abnormal operational states.

In the case of commercial design and analysis software, V&V activities associated with this type of software are performed primarily by the developer and/or the software service supplier. Additional V&V may be necessary for this type of software depending on the specific use and application.

As indicated above, the validation process relies on evaluating software as a solution to the right problem. Validation plays an especially important role for scientific and engineering software that use computational models to understand complex real-world phenomena and engineered systems. Validation techniques are employed to ensure that the model is a reasonably accurate representation that can be used for solving the problem for which it was designed with a high degree of confidence. Model validation plays a primary role in the overall software validation process for this type of software. The V&V of models, with an emphasis on model validation, is covered in Section 6.3 in the context of model development and evaluation.

5.2.8.1 Grading V&V

Safety Software:

This work activity applies to Level A and B custom-developed, configurable and acquired safety software, except for applications with utility and commercial design and analysis software. Under this work activity, custom-developed software will most likely have a larger number of and more detailed deliverables than others. For Level A safety software, all deliverables should be reviewed using V&V methods and traceability of the requirements should be performed. For Level A and B safety software, deliverables that include requirements, test plans and procedures, and test results should be reviewed using V&V methods.

[48] ASME NQA-1-2008/NQA-1a-2009, op. cit.,Part I, Requirement 11, Section 400.
[49] ASME NQA-1-2008/NQA-1a-2009, op. cit.,Part II, Subpart 2.7, Section 404.

For all Level A safety software except utility calculations, acceptance testing work activities should be planned and documented; acceptance test cases and procedures including expected results should be created; test results should be documented; and all test activity deliverables should be under configuration control. Level A utility calculations and Level B and C custom-developed, configurable, and acquired software may use a graded approach by applying less formality in the documentation. Simple checklists for acceptance test cases and procedures may be used in place of more detailed test cases and procedures. Test results should be documented and all test activity deliverables placed under configuration management

From a SQA perspective, configurable software should generally be treated the same as custom software. With configurable software, however it is important to differentiate between QA of the algorithms, macros, and logic that perform the calculations and QA of the configurable software itself (i.e., spreadsheet). Modern utility calculation applications such as spreadsheet programs have grown immensely in power and scope, with a corresponding growth in complexity. With complexity comes the risk that the program will not perform correctly if not managed under an acceptable SQA program. The addition of macro programming languages and the ability to incorporate “add-in” programs provide users with nearly the same capabilities as code developed with more traditional programming tools. Utility calculation applications are installed on virtually every desktop, and files containing algorithms and data can be easily modified by users. Consensus standards [50] provide guidance on appropriate V&V programs for utility calculations. Calculations performed using applications such as commercial spreadsheet programs may be treated in one of two ways:

- For relatively straightforward calculations, the results may be checked and verified in the same manner as a hand calculation.
- For more complex cases not amenable to simple methods, the user files containing the calculation formulas, algorithms, or macros should be subjected to the entire software life-cycle process. This approach may also be expedient for calculation applications that are reused frequently.

OSW:

For safety-affecting, custom-developed or configurable software with very high consequences of software failure, the V&V work activity described in this section should be implemented. For critical and general software, the V&V effort may include simplified result-oriented testing programs and less formality in documentation. The user organization may determine the level of the effort that is to be applied commensurate with the risk associated with the specific application.

[50] ASME NQA-1-2008/NQA-1-2009, op. cit., Subpart 4.1, Section 101.1.

5.2.9 Problem Reporting and Corrective Action

The problem reporting and corrective action process should be consistent with the site's QAP corrective action system and the adopted QA consensus standard. The scope of the process should extend to: (1) methods for documenting, evaluating and correcting software problems, (2) an evaluation procedure for determining whether a reported problem is indeed a defect or an error,[51] and (3) management roles and responsibilities for disposition of problem reports.[52]

Maintaining a robust problem reporting and corrective action process is vital to maintaining the reliability of the software. This process should not be separate from an existing reporting process provided it adequately addresses the items in this work activity.[53]

5.2.9.1 Reporting Problems by Suppliers and Developers

Procurement documents should identify the requirements for suppliers or developers of the software to report problem to the purchasers. The documents should also state the methods to be used for reporting, and the required response from the supplier or developers.[54]

5.2.9.2 Grading for Problem Reporting and Corrective Action

Safety Software:

For safety software other than safety system software, if the noted problem is an error, the problem reporting and corrective action process should identify the potential impacts of the error on past and present usage of the software, identify the impact of corrective action on previous development activities, appraise users of the problem report and identify strategies for avoiding the error pending implementation of the corrective action.

For errors in safety system software, the problem reporting and corrective action system should identify the potential impacts of the software error on past and present operations of the SSC and its controls, determine corrective action and mitigation strategies as necessary, and determine if the error has any impacts on the safety basis.

OSW:
For errors in safety-affecting software the problem reporting and corrective action system should identify the potential impacts of the software and determine corrective action and mitigation

[51] Software errors should be differentiated based on their significance and severity to degrade software function and system reliability. Software errors can be related to requirement errors, design errors, algorithmic processing errors, interface errors, performance errors and documentation errors.
[52] ASME NQA-1-2008/NQA-1-2009, op. cit., Part II, Subpart 2.7, Section 204
[53] ASME NQA-1-2008/NQA-1-2009, op. cit., Part IV, Subpart 4.1, Section 204
[54] ASME NQA-1-2008/NQA-1-2009, op. cit., Part II, Subpart 2.7, Section 301

strategies as necessary, and determine if the error has any impact on the safety basis. If the noted problem is a software error in a critical or general software application, the problem reporting and corrective action system should be implemented to address the appropriate requirements of the QAP.

5.2.10 Training of Personnel in the Design, Development, Use, and Evaluation of Software

A trained and knowledgeable staff is essential to assess and evaluate the SQA requirements to ensure the proper levels of quality and safety exists in the software. Training personnel in designing, developing, testing, evaluating, or using software application is critical for minimizing potential software failures. Although other SQA work activities may indicate that the software satisfies its operational objective, improper or invalid use of the software may negate the safety mitigation strategies included within the software.

Personnel who have direct responsibility for managing staff who design, develop, use, and evaluate software should receive specialized training on how these software-related activities should be managed and implemented. This training should be commensurate with the scope, complexity, and importance of the software. Completion of this management training should be documented and reviewed in the same manner as is performed.

Training may be necessary for the analysts, development and test teams, application users, and operations staff. The analysts and developers may have to be trained in fault tolerant methodologies, safety design methodologies, user interface design issues, testing methodologies, or configuration management to ensure delivery of a robust software application. Also, the software application users and operations staff may have to be trained to ensure that proper data are entered, that proper options and menus are selected, and that the results of the software can be interpreted and correctly applied.

Training should be commensurate with the scope, complexity, and importance of the tasks and the education, experience, and proficiency of the individual. Training and qualification requirements should meet the consensus standard(s) specified in the QAP. Personnel should also participate in continuing education and training as necessary to improve their performance and proficiency and ensure that they stay up-to-date on changing technology and new requirements.

Completion of training, education, and/or qualification requirements for staff involved in the development, testing, use, and evaluation of safety software should be documented and periodically reviewed. For OSW, similar training, education, and/or qualification for staff should be implemented and document in accordance with the QAP.

6. SPECIAL TOPICS SUPPORTING SOFTWARE QUALITY ASSURANCE

This section of provides information on:

- Model Development and Evaluation,
- Digital System Software Quality Assurance,
- Commercial Grade Dedication of Computer Programs, and
- Software Security Assurance

These topics were not addressed in the previous version of this Guide.

6.1 MODEL DEVELOPMENT AND EVALUATION

6.1.1 Introduction

Models are simplifications of the real world constructed to gain insights into select attributes of a particular physical, biological, economic, engineered, or social system. To create a model, scientific and engineering software is often used to simulate complex static and dynamic behavior observed in real-world natural phenomena and engineered systems. This section focuses on computational models that use measurable and estimated variables, numerical inputs, and mathematical relationships to produce quantitative outputs.[55]

The role of computational models is growing in decision-making despite their inherent limitations. Models, though generally subject to aleatory and epistemic uncertainties, can still provide valuable insights into facets of the real world. As a whole, models serve primary purposes of diagnosing observed phenomena or processes and predicting outcomes. Results from computational models aid the decision-making process by replacing or supplementing physical testing results in situations in which physical testing is impractical or physically impossible.

The next three subsections describe: (1) model development; (2) model V&V; and, (3) model evaluation. The development, V&V, and evaluation of models are closely tied to one another and are often executed in parallel through an iterative process. For example, the results from V&V and evaluation activities often provide insights for improvements to models that lead to further development. Since these topics can only be briefly discussed within the context of this guide, a basic summary is provided of concepts, processes and activities related to model development, V&V, and evaluation. Additionally, key principles are explained and selected sound practices from the published literature are highlighted.

[55] EPA/100/K-09/003, *Guidance on the Development, Evaluation, and Application of Environmental Models*, March 2009.

Following established practices from authoritative, peer-reviewed publications promotes consistent and informed management of models. Standards, guides, and publications from commercial industries, government agencies, and recognized subject matter experts offer recommendations for the systematic development and evaluation of models. Information summarized in the following subsections is derived from more than a dozen authoritative publications dealing with computational models. These publications are referenced in Appendix G of this Guide and should be consulted for further exploration of the particular topics.

6.1.2 Model Development

The development of models shares many features with development of software of other types. As shown in Figure 6.1-1, development typically starts with a conceptual model configured for the problem to be solved. The conceptual model is then translated into mathematical terms based on science and engineering principles. This process leads to a computational code which may contain scientific and/or engineering equations describing the process, phenomenon, or system of interest.

When equations describing the mathematical model cannot be solved analytically, a numerical solution technique such as finite element and finite difference is implemented. The numerical algorithm and its input parameters represent the computational model. The algorithm is then converted into computer code.[56]

[56] ANSI/ANS 10.7-2013, *Non-Real-Time, High-Integrity Software for the Nuclear Industry—Developer Requirements*

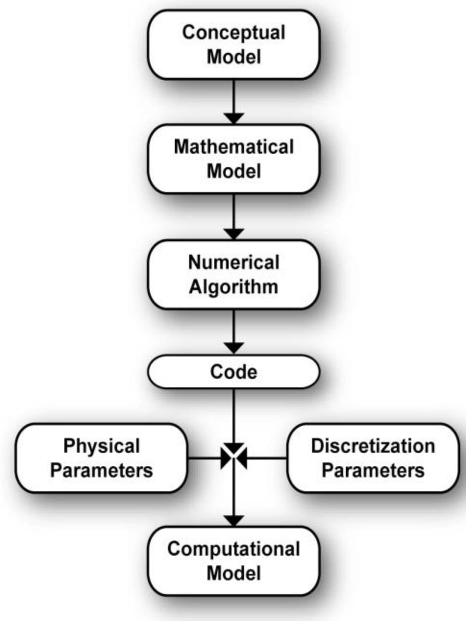 Conceptual Model Mathematical Model Numerical Algorithm Code Physical Parameters Discretization Parameters Computational Model	<u>Conceptual Model</u> Collection of assumptions, algorithms, relationships, and data that describe the reality of interest. <u>Mathematical Model</u> Mathematical equations, boundary values, initial conditions, and modeling data needed to describe the conceptual model. <u>Computational Model</u> Numerical implementation of the mathematical model, usually in the form of numerical discretization, solution algorithms, and convergence criteria.
Reproduced from ASME V&V 10-2006, Guide for Verification and Validation in Computational Solid Mechanics	[Definitions from LA-14167-MS, “Concepts of Model Verification and Validation,” Thacker et al., 2004.]

Figure 6.1-1 Framework for computational model development

The National Research Council advocates the use of a modular design in the computational model. A modular design allows creation of models of various complexities by adding or removing modules from the basic code structure. An additional benefit of a modular design is that debugging and testing are easier to perform.

6.1.3 Model Validation

Validation of the conceptual model is performed as shown in Figure 6.1-2. Note that the inner solid black arrows represent model and software development activities. The outer dashed red arrows represent V&V activities; arrow heads at each end signify an iterative process: V&V uncovers model deficiencies, the software developer corrects them and another cycle begins. This cycle repeats until software functionality meets the acceptance criteria for that phase.

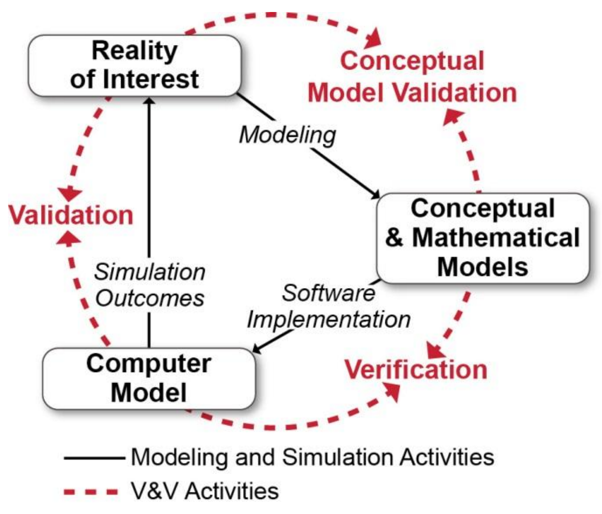

Figure 6.1-2 Model development and its V&V Processes [modified from Thacker, 2004 and Schlesinger, 1979][57]

Conceptual model validation evaluates the reasonableness and correctness of the model as it relates to its intended purpose, taking into account aleatory and epistemic uncertainty. The primary validation techniques used for these evaluations are face validation and trace analysis.[58] Data that are incorporated into the model can also be validated at this stage. Monte Carlo and Latin Hypercube sampling techniques can be employed to provide a series of outcomes at various probability levels for more comprehensive modeling applications. For these types of codes, care must be taken to ensure that the sampling technique is statistically sound and the Monte Carlo error is minimized. Larger samples generally reduce the size of the Monte Carlo error.

Documentation for the conceptual and mathematical models may include descriptions of assumptions, equations, algorithms, internal data, and application domain limitations related to hypotheses, theories, logic, interfaces, and solution approaches[59]. This documentation is reviewed for clarity, completeness and correctness as part of the verification process. Subject matter experts may be used for model verification.

[57] Thacker, B. H., S. W. Doebling, F. M. Hemez, M. C. Anderson, J. E. Pepin, and E. A. Rodriguez, *Concepts of Model Verification and Validation*, LA-14167-MS, Los Alamos National Laboratory, October 2004.
[58] Sargent, R. G., *Verification and Validation of Simulation Models*, In Proc. 2011 Winter Simulation Conf., ed. S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers Inc., 2011.
[59] Chew, Jennifer, and Cindy Sullivan, *Verification, Validation, and Accreditation in the Life Cycle of Models and Simulations*, Proceedings of the 2000 Winter Simulation Conference, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, Orlando, FL, December, 2000.

More complete and comprehensive V&V is performed after the model is translated into computer code. Code walkthroughs or inspections of key sections may be used to confirm that the mathematical model and algorithm are correctly represented throughout the application domain and that boundary conditions do not introduce computational errors.

Table 6.1-1 shows some of the commonly-used model validation techniques established in consensus standards and SQA guidance documents. The most reliable approach is comparison of software model output to experimental or field data when such data are both applicable and available. The next preferable technique is comparison against output from similar software that has undergone V&V. Other somewhat less reliable methods may provide either supplemental evidence of fidelity to real-world representativeness or an alternative means for model validation for situations in which real-world data are unavailable or are temporally or spatially limited.

Table 6.1-1 Model Validation Techniques[60, 61, 62, 63, 64, 65]

Model Validation Technique	Description
Comparison Against Field or Laboratory Data (also referred to as Output Validation)	Output from software model is directly compared to real world phenomena or controlled laboratory conditions.
Turing Tests	A panel of SMEs are presented with two blind sets of data; one representing the model results and the other representing real-world data, and asked to differentiate between them.
Comparison Validation (also referred to as Code-to Code Comparisons, Comparison to Other Models, Benchmarking)	Output from software model is directly compared to previously accredited software models dealing with a similar engineered system or similar real-world phenomenon.
Unit Testing (also referred to as, or closely related to, Functional Decomposition Validation, Piecewise Validation, Sub-model/Module Testing)	Output from simplified cases or individual sub-models are compared to: (1) available representative experimental field or laboratory data; (2) output from other representative software; or, (3) results from hand calculations based on applicable first principles or empirical evidence. Subsequent integration testing of each component then evaluates how well the individual components or modules are integrated.
Trace Analysis	The behaviors of different types of specific entities in the model are traced through the model to determine if the model's logic is correct.
Visualization Analysis (also referred to as Graphical Analysis, Animation Analysis)	Graphical results or output animations for representativeness to real-world behavior in terms of temporal variations, spatial distributions, correlated dependencies, and any other relevant observable trends are examined.
Sensitivity Analysis	Key input parameters are systematically varied over ranges of interest. The corresponding output from the software model is examined to evaluate consistency of the model response with respect to expected behavior.
Face Validation (also referred to as Peer Review)	A panel of SMEs critically reviews software model output and provides technical opinions on its reasonableness based on its knowledge of first principles and related empirical evidence. Visualization analysis and/or sensitivity analysis may be used whenever suitable data exist.

[60] ANSI/ANS 10.7-2013, *Non-Real-Time, High-Integrity Software for the Nuclear Industry—Developer Requirements*

[61] Sargent, R. G., *Verification and Validation of Simulation Models*, In Proc. 2011 Winter Simulation Conf., ed. S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers Inc., 2011.

[62] ATEC, *Test and Evaluation, Modeling and Simulation Verification, Validation, and Accreditation Methodology*, Pamphlet 73-21, Department of the Army, Army Test and Evaluation Command, Alexandria, VA, April 2007

[63] M&SCO (2006), *VV&A Recommended Practices Guide*, Modeling and Simulation Coordination Office (M&S CO), *VV&A Recommended Practices Guide*, September 2006.

[64] DA, *Verification, Validation, and Accreditation of Army Models and Simulations*, Pamphlet 5-11, Department of the Army, Washington, DC, September 1999.

[65] NIST, *Reference Information for the Software Verification and Validation Process*, NIST Special Publication 500-234, National Institute of Standards and Technology, Gaithersburg, MD, March 1996.

6.1.4 Model Selection and Evaluation

Model evaluation gauges whether a model applies to a particular class of problems, taking into account inherent limitations of the model and aleatory and epistemic uncertainties in the results.[66,67,68,69,70] One often-overlooked type of uncertainty is that there may be more than one model suitable for a specific problem. A sound practice to address this uncertainty is to document the basis for selecting the model over any of the alternatives.[71]

A guiding principle for model selection is to choose the simplest model suitable for the intended purpose.[72] A set of best practices has been developed based on this principle.[73] Figure 6.1-3 illustrates the principle of finding an optimal level of complexity for models having two primary sources of uncertainty:[74]

- Model framework uncertainty, which is a function of the soundness of the model's underlying scientific foundations; and
- Data uncertainty, which arises from measurement errors, analytical imprecision and limited sample size during collection and treatment of the data used to characterize the model parameters.

Simple models, represented on the left side of graph, have a high level of total uncertainty driven by framework uncertainty. This framework uncertainty arises from incomplete modeling of real-world phenomena. Data uncertainty is low because only a few input variables are needed to run simple models. Complex models, represented on the right side of graph, also have a high level of total uncertainty composed mainly of data uncertainty. The data uncertainty arises from the large set of input variables needed to "feed" the numerous equations used in complex models. The optimal level of complexity is located at the center or the graph, where the total uncertainty reaches a minimum level owing to a balanced treatment of the model framework and data uncertainties.

---

[66] Thacker, B. H., S. W. Doebling, F. M. Hemez, M. C. Anderson, J. E. Pepin, and E. A. Rodriguez, *Concepts of Model Verification and Validation*, LA-14167-MS, Los Alamos National Laboratory, October 2004.
[67] Overcamp, William L. and Timothy G. Trucano, *Verification and Validation in Computational Fluid Dynamics*, SAND2002-0529, Sandia National Laboratories, March 2002.
[68] Roach, Patrick J. Verification and Validation in Computational Science and Engineering, Hermosa Publishers, Albuquerque, NM, 1998
[69] EPA/100/K-09/003, op. cit.
[70] National Research Council, *Models in Environmental Regulatory Decision Making*, National Research Council, Washington D.C., 2007
[71] National Research Council, op. cit.
[72] National Research Council, op. cit.
[73] EPA/100/K-09/003, op. cit.
[74] EPA/100/K-09/003, op. cit.

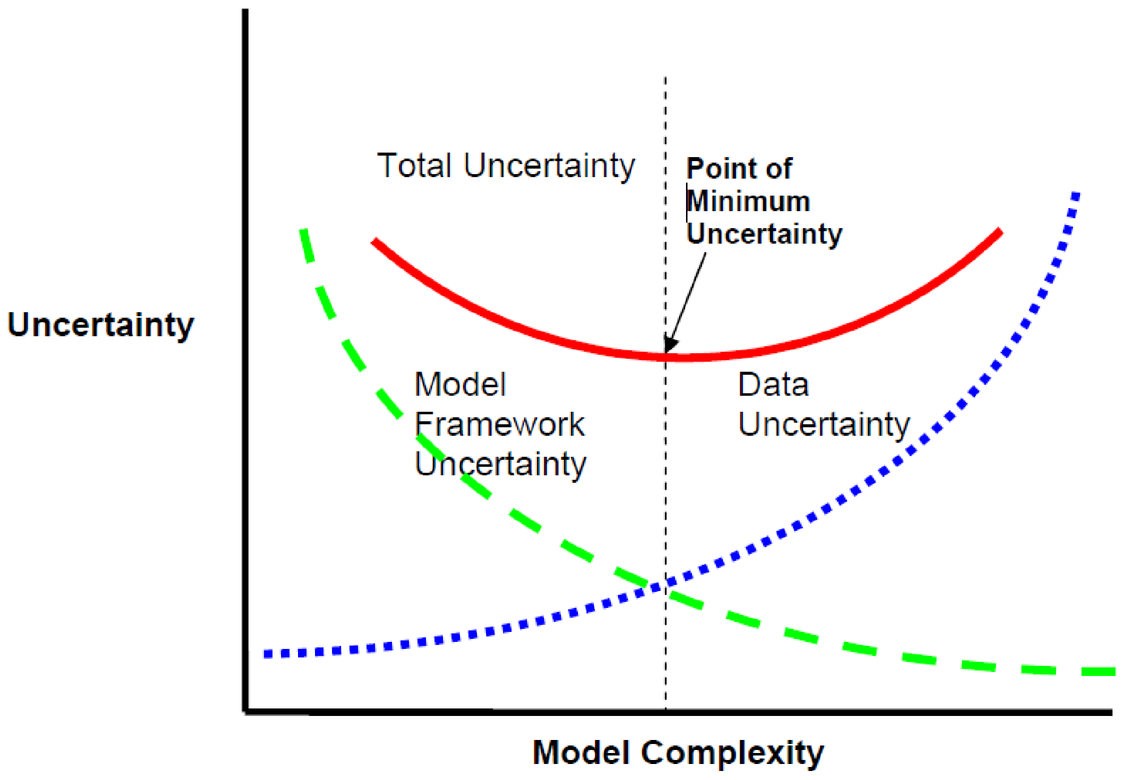

Figure 6.1-3 Optimal Level of Model Complexity
[Reproduced from EPA, 2009; adapted from Hanna, 1988]

Since different models contain various types and ranges of uncertainty, sensitivity analyses can be useful early in the development phase to quantify the relative importance of each parameter to the expected outcome. Sensitivity analysis is the process of determining to what extent changes in the model input values or assumptions, including boundaries conditions and model functional form, affect the model outputs. Model complexity can be constrained by eliminating parameters when sensitivity analyses show that they do not significantly affect the outputs and there is no process-based rationale for including them.

With the increasing use of advanced computer models such as computational fluid dynamic (CFD) models, uncertainty quantification has gained considerable attention in the published literature.[75, 76, 77, 78, 79] Uncertainty quantification of the model often requires advanced

[75] Thacker, op. cit.
[76] Overcamp, op. cit.
[77] Roach, Patrick J. *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, NM, 1998
[78] EPA/100/K-09/003, op. cit.
[79] National Research Council, op. cit.

mathematics such as Monte Carlo and Latin Hypercube simulations and Bayesian analysis. Significant sources of uncertainty and their effects on model predictions should be documented to the extent achievable.

Exact quantification of uncertainty is not feasible in some situations, especially with highly complex models. In such cases, a qualitative approach using peer review and other V&V tests may be the only realistic option. [80] In qualitative analysis, the directions of the output behaviors are examined and also possibly whether the range of magnitudes is within reasonable bounds. In quantitative analysis, both the directions and the precise magnitudes of the output behaviors should be examined. SMEs on the system usually know the directions and often know the general values of the magnitudes of the output behaviors.[81]

6.2 DIGITAL SYSTEM SOFTWARE QUALITY ASSURANCE

This section provides SQA guidance specific to digital systems and system components. This guidance should be considered in conjunction with Sections 5.1 and 5.2 of this Guide. The definition of "digital" to be used here is found in IEEE-STD-24765, *Systems and Software Engineering Vocabulary*, "pertaining to data that consists of digits as well as to processes and functional units that use the data."

SQA for digital systems is an expanding and complex area. Among the available references on this topic is DOE-STD-1195, *Design of Safety Significant Safety Instrumented Systems Used at a DOE Nonreactor Nuclear Facilities.* A particularly useful tool in this Standard is the crosswalk of recommended implementing standards for each of the SQA work activities specified in DOE O 414.1D. Acceptable industry references found in this crosswalk include ANSI/ISA 84.00-01, *Functional Safety: Safety Instrumented Systems for the Process Industry Sector*; and ANSI/IEEE STD 7-4.3.2, *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations.*

The emphasis of this section of the Guide is on SSS that performs monitoring or control functions. However, this guidance may also be considered for other applications using a graded approach.

6.2.1 Digital System Identification

Identifying digital system software is an important step in digital system SQA. In many systems, such as programmable controller-based facility control systems, the identification and use of

[80] ANSI/ANS 10.7-2013, *Non-Real-Time, High-Integrity Software for the Nuclear Industry – Developer Requirements*, ANSI/ANS, March 2013.
[81] Sargent, op. cit.

digital software is apparent. However, in other systems it may not be readily apparent whether the device relies on software many devices on the market today that did not previously use software now use it.

To illustrate: in November 1993, a power station's load sequencer for an emergency diesel generator failed to automatically load safety-related equipment onto the emergency bus.[82] The incident occurred following replacement of electromechanical time/relays with microprocessor-based time/relays. Inadequate qualification and commercial grade dedication of the new digital component led to the failure. In July 2010, a failure occurred in a safety-significant PLC processor module for a tritium air monitor. It was later discovered that notifications of defective firmware identified some, but not all, of the defective processors. The defective firmware caused the incident.[83]

Because of the increase of such components in the nuclear industry, industry is using the term "embedded digital device" (EDD). Such a device consists of one or more electronic parts that (a) require the use of software, software-developed firmware, or software-developed logic, and (b) is integrated into equipment to implement one or more system safety functions. EDDs include digital components with executable code or software-developed logic that is permanent or semi-permanently installed within the device. EDDs include programmable logic devices (PLD), field programmable gate arrays, application-specific integrated circuits, erasable programmable read-only memory, electrically erasable programmable read-only memory, and complex (PLD).

6.2.2 SQA for Digital System Design

As referenced in DOE-STD-1195, ANSI/ISA-84.00.01 is a valuable reference digital system design. Key digital system design SQA for digital systems design should include these concepts:

- Isolation: Critical components should be separated from each other to preclude undefined and/or unintended interactions. Options include encapsulation, information hiding, and formal interfaces to prevent unintended software execution or malfunction. Safety systems should be separated from non-safety systems where possible. Barrier requirements should be used to prevent non-safety functions from interfering with safety functions.
- Independence/Diversity: Independent, diverse systems should be employed where the stimuli originate from and are handled by separate components with different designs, independent hardware inputs and independent software modules. Note that because

[82] U.S. Nuclear Regulatory Commission Information Notice No. 94-20, "Common-cause Failures due to Inadequate Design Control and Dedication," March 17, 1994
[83] U.S. Department of Energy, Office of Health, Safety and Security Safety Advisory for Software Quality Assurance, Advisory No. 2010-08, "Firmware Defect in Programmable Logic Controller," October 2010.

different manufacturers may use the same processor or software, common mode failures may not be averted simply by buying from several component manufacturers.

- Fail-Safe Design: Component should be designed to become inoperable in a safe mode without compromising isolation features.
- Incompatibility/Longevity: Systems should be designed to ensure integrated compatibility throughout the life-cycle. Note that replacement components such as sensors, logic devices, and control devices may contain upgraded software or require upgraded support software. In some cases, upgrades may introduce new embedded software that was not in the original component. The replacement software/hardware should be compatible with existing software and hardware.
- Multiple/Common-Cause Failures Multiple/common-cause failures should be evaluated in the integrated system hazard analysis. The design should prevent such failures.

6.2.3 SQA for Digital System Acquisition

In DOE facilities, many digital systems are acquired from suppliers. The guidance below for digital systems should be considered for the acquisition of items or services.

6.2.3.1 Software Identification

The purchaser should attempt to determine the presence of software in components and ensure compatibility of software found with existing hardware, software and environmental conditions. Procurement documentation and specifications should require the supplier to state whether software is embedded in the procured items. If so, the supplier should have to document the software sufficiently to support commercial grade dedication (as necessary), operation and maintenance.

6.2.3.2 Commercial Grade Dedication (CGD)

The acquisition of commercial grade items for digital systems should be consistent with the CGD process in Section 6.3.

6.2.3.3 Electromagnetic Compatibility (EMC)

The EMC of original equipment may not support reliable operation of new equipment containing EDD. Conversely, the EMC characteristics of new equipment containing EDDs may be insufficient to support continued, reliable operation of nearby unmodified equipment. In both cases, the digital systems should be evaluated and tested in the intended environment (or simulated environment) prior to placing them into service.

6.2.3.4 Pre-Operational Testing

Many digital system acquisitions are replacement of components in existing, operating systems. Acquisitions should therefore emphasize testing prior to placing the digital component/system in operation. Testing should start with off-site bench/factory testing followed by on-site testing, preferably in actual operating environments or if not possible, in a simulated operating environment. For onsite testing of systems that will eventually use hazardous materials, onsite tests should be performed prior to introduction of the material ("cold-tested").

6.2.4 SQA for Digital System Use and Maintenance

Consider these factors when establishing SQA requirements for use and maintenance of digital systems:

- Configuration management should be used for the digital system's component software and hardware to promote system compatibility.
- Controls of both physical and electronic access to the digital system and data should be used to prevent unauthorized access and secure the system from electronic vulnerabilities.
- Training of engineers and operations and maintenance personnel should be provided on digital system SQA; training should include awareness of potential EDD in new systems or replacement components in existing systems.
- Well-designed, periodic in-use tests and assessments should be performed to identify issues, including those that may arise from unintended cumulative effects of safety or interconnected digital system changes.
- Controlled and integrated software and hardware retirement should be used to ensure that compatibility is retained among all systems.

6.3 COMMERCIAL GRADE DEDICATION OF COMPUTER PROGRAMS

6.3.1 Introduction

Commercial grade dedication (CGD) consists of qualifying for use items not designed, developed, or manufactured in accordance with the requirements of ASME NQA-1. CGD is part of the acquisition process, used to provide reasonable assurance that the computer program or service will perform its intended safety function in spite of pedigree deficiencies.

For computer codes, CGD involves comparing intended functions with the supplier's design to verify that the code meets functional requirements, critical characteristics, and established acceptance criteria. Detailed requirements of the CGD process can be found in various versions

ASME-NQA-1.[84,85] Additional guidance is available in Electric Power research Institute (EPRI) TR-1025243[86], EPRI TR-106439 [87] and DOE's *Office of Environmental Management Guidance for Commercial Grade Dedication*, April 2011. This section highlights some of the important concepts used in performing CGD for computer programs.

The determination of critical performance characteristics is essential to the CGD process. This determination is normally made by the organization responsible for the CGD. The dedicating entity can be the manufacturer, a third-party organization, the purchaser, or user facility technical staff.

6.3.2 Approach

Computer program dedication proceeds in two steps, termed technical evaluation and acceptance. During the technical evaluation phase, CGD documentation is developed to establish the approach, anticipated activities, and methods. During the acceptance phase, specified methods of verification are applied to ensure that the critical characteristics for acceptance (CCFA) meet the specified acceptance criteria. Acceptance is documented by the individual who performed the verification.

6.3.3 Dedication of Commercial Grade Computer Programs

Organizations planning to use procured COTS computer codes not developed to ASME NQA-1 requirements should perform the following tasks to meet CGD requirements:

- Determine the safety function that the computer program is has to perform;
- Determine the capabilities, application domain, and limitations for the program's intended safety function;
- Identify and document critical characteristics and acceptance criteria;
- Select the dedication method; and,
- Generate instructions for use within the limits of the dedicated capabilities.

[84] ASME NQA-1-2008/NQA-1a-2009, op. cit., Subpart 2.7 and Subpart 2.14
[85] ASME NQA-1-2012 Subpart 2.7, Subpart 2.14 and Subpart 3.2-2.14
[86] EPRI TR-1025243, *Plant Engineering: Guideline for the Acceptance of Commercial-Grade Design and Analysis Computer Programs Used in Nuclear Safety-Related Applications*.
[87] ERPI TR-106439, *Plant Engineering: Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications*.

6.3.4 Technical Evaluation and Acceptance

6.3.4.1 Technical Evaluation Phase

In the technical evaluation phase, all safety functions should be identified and specified in the CGD documentation. Safety functions are defined as functions the computer program performs to ensure safety. The safety function may be, and often is, a subset of the computer program's overall functions. The critical characteristics to be satisfied for acceptance and the methods for verifying acceptance should also be identified in the CGD documentation.

Commercial grade computer codes can have numerous design characteristics that are related to composition, identification, or performance and may or may not impact a safety function. Item characteristics include product identification characteristics and other characteristics that are inherent to the item's design but are not required or used in the purchaser's application to support the safety function. The critical characteristics for design (CCFD) are those design characteristics that are important to the performance of the computer code that allows it to perform its safety function. However, it is not necessary to verify all design characteristics and/or CCFD to provide reasonable assurance that the computer code will perform its intended safety function. CCFA should be identifiable and measurable attributes of the computer program that, when verified, will provide reasonable assurance that the computer program will perform the intended safety function when required. Reasonable assurance is considered to have been provided when, in the opinion of the responsible engineer, a sufficient number of CCFD and computer program characteristics have been verified and documented as CCFA to provide sufficient evidence that the computer program will be capable of performing its safety function.

Critical characteristics fall into five categories:

- Computer program characteristics that uniquely identify the computer code being dedicated;
- Host system characteristics of the host hardware and computer code operating environment required for code execution;
- Interface characteristics of the inputs, outputs and user interface;
- Computer program characteristics of the computer program relative to performance, failure modes and functionality; and
- Vendor characteristics for vendor support, vendor qualifications and vendor quality assurance program.

CGD documentation should identify the cognizant design authority assigned to develop CGD documentation, perform the technical evaluation, generate the Requirements Traceability Matrix, and identify critical characteristics in the host system and interfaces.

6.3.4.2 Acceptance Phase

Acceptance should be based on documented acceptance criteria, supplier information, quality history, and degree of standardization. There are four methods, identified below, which should be used to verify that critical characteristics have been met and that the computer code meets all requirements. At least two or more of these methods should be used as specified in the CGD documentation to verify the acceptance criteria of each critical characteristic.

Method 1 - Special Tests, Inspections, or Analyses

Special tests, inspections, or analyses should be conducted either individually or in combination before or after receipt of the computer code, to verify conformance with the acceptance criteria. Method 1 is particularly useful when:

- Critical characteristics can be verified with tests/inspections;
- Data to verify critical characteristics is available in existing documents such as specifications, drawings, computer code life cycle documents, instruction manuals, and catalogs;
- The computer code does not include functionality beyond the safety functions; or
- Post-installation tests in the environment can be conducted.

Method 2: Commercial Grade Survey

A commercial grade survey assesses a supplier’s procedures and controls related to the program’s critical characteristics. The survey should be performed at the supplier’s facility and should be completed prior to issuing the purchase order for the computer program. A survey of a supplier may be appropriate under the following circumstances:[88]

- The supplier/manufacturer has implemented what appears to be appropriate, documented controls over the critical characteristics;
- Multiple computer code items are frequently procured from the same supplier; or
- Critical characteristics are not easily verified after receipt.

Method 3 - Source Verification

Source verification is a method of acceptance conducted at a supplier's location during program development. Source verifications should include activities such as witnessing the development

[88] DOE, *EM Guidance for Commercial Grade Dedication*, April 2011, Section 2.2.2

of the computer program, performance tests, or final inspections, as applicable. It should also include verification of the supplier's design and the implementation of process controls as applicable to the identified critical characteristics. Source verification may be appropriate under the following circumstances:

- In-process verification of one or more critical characteristics is needed;
- Non-conformances have been detected during prior receipt inspections;
- Problems or deficiencies are known to exist in the supplier's QA program; or
- Computer code being procured is the first of its kind being developed.

Method 4 - Acceptable Supplier Item or Service Performance Record

This method of acceptance is based on the demonstrated and documented past performance of a supplied, similar computer program over a period of time. This method of acceptance should be supported by the use of one of the other three methods. Method 4 should not rely on a single source of information, and should include a large data set of successful historical performance of the computer program. Method 4 may be appropriate in the following circumstances:

- Critical characteristics are not easily verified after receipt;
- It can be verified that the performance data used is directly applicable to the verification of critical characteristics specific to the intended application;
- The performance record is from similar conditions of service, environmental conditions, failure modes, maintenance programs, testing, or other conditions equivalent to the intended application of the computer program; or
- Monitored performance of the computer code has been installed and operated in a similar environment as the intended facility.

6.4 SOFTWARE SECURITY ASSURANCE

6.4.1 Introduction

Software security assurance provides a level of confidence that software is free from exploitable design vulnerabilities. Potential software vulnerabilities include those maliciously inserted into the software and those inadvertently inserted during the software life cycle. Assurance is achieved by planned, systematic processes and assessments to confirm compliance with design requirements for software security.

Secure software is designed, configured, implemented, and supported so that it can: 1) continue operating correctly in the presence of most attacks, by either resisting exploitation of weaknesses or by tolerating the errors that result from such exploits; and 2) limit the damage resulting from any errors caused by attack-triggered weaknesses and recover as quickly as possible from such errors.

6.4.2 Methodology for Designing and Implementing Secure Software

Design and implementation of software should include establishing a software security assurance program that ensures secure software is properly developed and used throughout its life-cycle.

An effective software security assurance program should address the following areas:

- Use of secure software architecture/design objectives and design principles. These include:
  - Architectural design that eases creation and maintenance of an assurance case.
  - Architectural design that eases traceability, verification, validation, and evaluation.
  - Architectural design that eliminates possibilities for violations.
  - Architectural design that ensures certification and accreditation of the operational system.
  - Architectural design that provides predictable execution behavior.
  - Architectural design that avoids and/or works around any security-endangering weaknesses in the environment or development tools.
  - Minimization of the number of safety-related components to be trusted.
  - A system that is designed to do only what the specification calls for and nothing else.
  - A system that is designed to tolerate security violations.
  - A system that is designed to avoid security risks to other systems in the environment.
  - A system that is designed to survive potential attacks.
- Specification of software security design requirements that cover both overt functional security (e.g., use of applied cryptography) and emergent security properties, as revealed by abuse cases and attack patterns. These requirements should address all known threats applicable to the design and implementation processes for the software being developed or modified, such as those from off-shoring, outsourcing, and insiders.
- Establishing a software security specification that builds abuse cases that describe a software system's behavior when under attack, in order to clarify what areas and components of a software-based system should be protected from which threats and for how long. This specification should also: 1) describe potential protection measures against issues such as software development errors, 2) guide software security implementation, 3) assist in the monitoring of software security postures, and 4) help make software-based systems more adaptable to the changing landscape of software security.
- Enlisting system engineering techniques, such as defense-in-depth (DID) measures (e.g., application layer firewalls, XML security gateways, sandboxing, code signing) and secure configurations, with operational security practices, including patch management

and vulnerability management. Application security DID measures operate predominately by using: 1) boundary protections to recognize and block attack patterns and 2) constrained execution environments to isolate vulnerable applications, thus minimizing their exposure to attackers and their interaction with more trustworthy components. Operational security measures focus on reducing the number or exposure of vulnerabilities in applications by repeatedly reassessing the number and severity of residual vulnerabilities and the threats that may target and exploit them, so that DID measures can be adjusted to maintain their level of effectiveness.

- Development of a secure software configuration management (SSCM) system that establishes traceability of software development activities as part of the SCM system described in Section 5.2.3 of this Guide. This system should ensure that the management role for software development activities is separate from the roles for development or testing of software. At a minimum, the development artifacts of potentially high-consequence software should be treated as items subject to the SSCM system. The SSCM system should consider implementation of the following software security practices:
  - Access control for software development artifacts, including but not limited to threat models and use/misuse/abuse cases; requirements, architecture, and design specifications; source code and binary executables; test plans/scenarios/reports/oracles, code review findings, and vulnerability assessment results; installation/configuration guides, scripts, and tools; administrator and end user documentation; Independent V&V documents; and security patches and other fixes;
  - Time stamping and digital signature of all configuration items upon check-in to the SSCM system;
  - Base-lining of all configuration items before they are checked out for review or testing;
  - Storage of a digitally signed copy of the configuration item with its configuration item progress verification report;
  - Separation of roles/access privileges and least privilege enforcement, for SSCM system users;
  - Separation of roles and duties (developing, testing, etc.) within the software development team;
  - Authentication of developers and other users before granting access to the SSCM system;
  - Audit of all SSCM system access attempts, check-ins, check-outs, configuration changes, traceability between related components as they evolve, and details of other work done;
  - Flexible but carefully controlled delegation of SSCM administrator privileges;
  - No remote access, or remote access only via encrypted, authenticated interfaces;

    - Reporting of differences between security aspects of previous and subsequent versions and releases;
    - Ability to assign access permissions to users and to restrict access to the repository based on those permission assignments;
    - Ability to limit read and write accesses (check-ins and check-outs) to a single directory; and
    - Tracking for all fixes, patches, updates, and new releases by the suppliers of commercial off-the-shelf and open source software components.

- Writing procedures for the acquisition and/or development of software that include requirements for testing and analysis of software. These procedures have the goal of exposing and rectifying potential software security vulnerabilities prior to its use.
- Performing quality assurance of key software security activities throughout the software development life cycle in order to provide assurance that software security requirements are adequate. As a minimum, software security activities that should be subjected to QA reviews and controls are:
    - Assessing development risks (i.e., those related to running a development project, which would typically include risks associated with business requirements, benefits, technology, technical performance, costing, and timescale);
    - Ensuring that security requirements have been defined adequately;
    - Ensuring that security controls agreed to during the risk assessment process (e.g., policies, methods, procedures, devices or programmed mechanisms intended to protect the confidentiality, integrity or availability of information) have been developed and implemented;
    - Determining whether security requirements are being met effectively; and
    - Establishing security operations that monitor the behavior of a software system for indications of attacks and exploits against the software. Knowledge gained through monitoring attacks should be fed back into the activities described above and those listed in Section 5.6.3 of this Guide on testing and analysis of software.
- Utilization of software security training that provides stakeholders adequate security education in software development. Training concepts include security awareness, knowledge of attacks on previous related applications, understanding of attackers' interests, and knowledge of secure development practices.

6.4.3 Testing and Analysis of Software for Security Vulnerabilities

Software security testing and analysis should occur throughout the software development life cycle and should include activities such as:

- Static analysis/review of source code that uses static analysis tools to detect common vulnerabilities.

- Direct code analysis that extends static analysis by using tools that focus on verifying the code's overall conformance to a set of predefined properties, which can include security properties such as non-interference and separability, persistent bisimulation-based non-deducibility on compositions (BNDC), non-inference, forward-correctability, and non-deductibility of outputs.
- Source code fault injection, a form dynamic analysis in which the source code is "instrumented" by inserting changes, then compilation and execution of the instrumented code to observe the changes in state and behavior that emerge when the instrumented portions of code are executed.
- Fault propagation analysis that involves two techniques for fault injection of source code: extended propagation analysis and interface propagation analysis.
- Risk analysis of the software architecture and design that includes documenting assumptions and identifying possible attacks, and uncovering and ranking architectural flaws for mitigation. Recurrent risk tracking, monitoring and analysis should be ongoing throughout the software development life cycle.
- The use of software security checklists.
- Implementation of "canned" black box tests offered by automated application security and penetration tests with tools and solutions that are driven by the results of the architectural risk analysis.
- Risk-based security testing using standard functional testing techniques. Risk-based security testing of software relies on test scenarios that are based on attack patterns.
- Independent software security reviews which analyze, assess, and/or test the software's design and implementation.

6.4.4 Security Controls

Security controls for information systems and organizations consist of safeguards and countermeasures to: (a) protect the confidentiality, integrity and availability of information processed, stored and transmitted by those systems and organizations, and (b) satisfy a set of defined security requirements.

The types of activities an organization should include for such controls are:

- Identification of software types;
- Risk assessment of their systems to determine the security categorization;
- Selection of security controls required to mitigate risk throughout the system life-cycle;
- Determination of the required level of assurance needed for confidence that security controls are achieving intended goals; and
- Risk management and continuous monitoring and maintenance to adjust risk according to the changing threat landscape.

Pursuant to DOE O 205.1B, *DOE Cyber Security Program*, guidance for security controls for all U.S. federal information systems, except those relating to national security, can be located in National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53, *Security and Privacy Controls for Federal Information Systems and Organizations*. Guidance for security controls for industrial control systems is provided in NIST SP 800-82, *Guide to Industrial Control Systems Security*. The Committee on National Security Systems Instruction No. 1253, *Security Categorization and Control Selection for National Security Systems*, provides guidance on security controls for national security systems, with Attachment 1 providing guidance for industrial control systems.

Nuclear Regulatory Commission (NRC) Regulatory Guide 5.71, *Cyber Security Programs for Nuclear Facilities*, provides guidance on satisfying 10 CFR §73.54, *Protection of Digital Computer and Communication Systems and Networks*. There are also numerous industry best practices and guidance documents that address themes such as reference architectures, layered security, defense in depth, and product specific security features.

Security categorization examples for various types of software are provided in Tables 6.4-1 through 6.4-3.

Table 6.4-1, Security Categorization for Acquired Low-Impact OSW

Activity	Guidance
Assess risk, security categorization	NIST 800-37, Federal Information Processing Standard (FIPS) 199
Choose security controls and assurance level	FIPS 200, NIST 800-53 tailored baseline
Continuously monitor and maintain	NIST 800-137

Table 6.4-2, Security Categorization for Custom High-Impact OSW

Activity	Guidance
Assess risk, security categorization	NIST 800-37, FIPS 199
Choose security controls and assurance level	NIST 800-53, Committee on National Security Systems Instruction (CNSSI) 1253 tailored baseline
Continuously monitor and maintain	NIST 800-137

Table 6.4.-3, Security Categorization for Configurable High-Impact OSW

Activity	Guidance
Assess risk, security categorization	NIST 800-37, FIPS 199
Choose security controls and assurance level	FIPS 200, NIST 800-53 tailored baseline
Continuously monitor and maintain	NIST 800-137

7. ASSESSMENT

7.1 GENERAL

DOE assessment requirements in 10 CFR Part 830 Subpart A and DOE O 414.1D should be applied to software management and control issues.

7.2 DOE AND CONTRACTOR ASSESSMENTS

DOE assesses the effectiveness of its actions in resolving problems with software management and control. DOE also evaluates the adequacy and implementation effectiveness of DOE and contractor software management and controls.

Contractors are expected to assess their software management and controls in accordance with DOE O 414.1D and their QAPs.

A suggested model criteria review and approach document (CRAD) for safety software is provided in Appendix E. This model contains software qualification assessment criteria for assessing the safety software

APPENDIX A. MANAGEMENT OF SOFTWARE BY TEN QA CRITERIA

This Appendix discusses the applicability of the ten software quality assurance (SQA) work activities with respect to the ten quality assurance (QA) criteria as described in Department of Energy (DOE) Order (O) 414.1D, Quality Assurance.

This Appendix addresses the application of ten quality assurance (QA) criteria in Department of Energy (DOE) Order (O) 414.1D, *Quality Assurance* with respect to the ten software quality assurance (SQA) work activities described in this Guide. [89] However, the QA criteria are broad enough to cover several SQA work activities and only the most significant work activities have been identified.

Criterion 1 – Management/Program

(a) Establish an organizational structure, functional responsibilities, levels of authority, and interfaces for those managing, performing, and assessing the work,
(b) Establish management processes including planning, scheduling, and providing resources for the work.

SQA requirements are addressed in the contractor's quality assurance program (QAP), which may be a site-wide QA document or in other documents. The contractor's QAP should document methods for planning, performing, assessing, and improving the adequacy of software items and SQA work activities. Basic elements that should be included in an SQA program include:

- Organizational structure, roles and responsibilities;
- Software engineering methods including risk-based approach;
- Minimum required documentation;
- Standards, conventions, techniques, or methodologies for software development, acceptance testing, and compliance;
- Required software reviews; and
- Resource planning and scheduling.

For additional guidance, refer to Section 5.2.1, *Software Project Management and Quality Planning* and Section 5.2.2, *Software Risk Management.*

Criterion 2 – Management/Personnel Training and Qualification

(a) Train and qualify personnel to be capable of performing their assigned work.

[89] Adopted from SQASG-TP-10-01-REV. 1, *A Systematic Approach to Implementing the Quality Requirements of DOE O 414.1D for Software*

(b) Provide continuing training to personnel to maintain their job proficiency.

This criterion should be integrated into the QAP section on qualifications and training. Requisite qualifications and training should be established for all personnel associated with the development, acquisition, configuration, management, use, oversight, and retirement of software.

Software development personnel should be qualified to perform all software life-cycle activities either through prior experience, classroom training, or on-the-job training. Software training is especially important for developmental personnel to ensure that new software correctly handles real-world phenomena and engineering systems. SQA personnel should be qualified through experience and/or training to perform QA activities such as testing, verification, validation, and audits.

Software user training can be accomplished through classroom demonstrations, laboratory exercises, online help functions, and study of the user's manual. User training should describe features, functions, and limitations of the software and how to navigate through the program prompts. It should also include information on error messages resulting from improper input and user mistakes.

For additional guidance, refer to Section 5.2.10, *Training Personnel in the Design, Development, Use, and Evaluation of Software.*

Criterion 3 – Management/Quality Improvement

(a) Establish and implement processes to detect and prevent quality problems
(b) Identify, control, and correct items, services, and processes that do not meet established requirements.
(c) Identify the causes of problems, and include prevention of recurrence as a part of corrective action planning.
(d) Review item characteristics, process implementation, and other quality related information to identify items, services, and processes needing improvement.

Corrective action procedures should ensure that the causes of software problems are identified and corrective actions taken to:

- Identify, evaluate, document, and correct the problem;
- Assess problems for their impact on past and present uses of the software;
- Develop changes to software in accordance with established design controls; and
- Provide necessary information to affected users.

Software that does not meet its performance requirements, or produces errors or anomalies, should be withdrawn from use until such problems are resolved.

For additional information, refer to Section 3.2.2, *Continuous Improvement, Measurement, and Metrices,* Section 5.2.8, *Verification and Validation* and Section 5.2.9, *Problem Reporting and Corrective Action for additional guidance.*

Criterion 4 - Documents and Records

(a) Prepare, review, approve, issue, use, and revise documents to prescribe processes, specify requirements, or establish design.
(b) Specify, prepare, review, approve, and maintain records.

Activities that affect software quality should be performed in accordance with documented instructions, plans and procedures. The instructions and procedures should define the processes for performing these activities, establish requirements and responsibilities, and be reviewed and approved by management. Records that furnish documentary evidence of software quality should be maintained in a dedicated storage facility or embedded in a site-wide records management system.

For additional guidance, refer to Section 5.2.1, *Software Project Management and Quality Planning* and Section 5.2.3, *Software Configuration Management,*

Criterion 5 – Performance/Work Processes

(a) Perform work consistent with technical standards, administrative controls, and other hazard controls adopted to meet regulatory or contract requirements using approved instructions, procedures, or other appropriate means.
(b) Identify and control items to ensure proper use.
(c) Maintain items to prevent damage, loss, or deterioration.
(d) Calibrate and maintain equipment used for process monitoring or data collection.

Performance of Work

The first requirement in this criterion requires use of technical standards and other hazard controls to meet regulatory or contractual requirements. DOE O 414.1D requires the selection of a consensus standard or set of standards which should be used for performing the tasks associated with SQA work activities.

Additionally, instructions or procedures should be created and used for software development, acquisition, and configuration. These instructions and procedures should be reviewed and approved by management. The instructions and procedures should describe the applicable SQA work activities for safety and other software (OSW). The need for and the level of detail in such written instructions and procedures should be based on using a graded approach, on the safety and mission significance (e.g., grading levels), complexity of the task, and the development environment.

Identify and Control Items

Software configuration management should be the primary work activity for implementing this requirement. All software and associated hardware items should be identified and controlled. This requirement covers computer programs and associated software tools and system software, software development and testing procedures, user and other related documents, and data. Controls should be established to assure that only correct and accepted items are used and installed.

Prevent Damage, Loss or Deterioration of Items

Controls should be established for protecting software from unauthorized access or inadvertent damage or degradation during all phases of the software life-cycle. These controls should address both the security of the computer system and the critical data that reside on that system. Processes should also be defined to identify the media to be controlled for each software product, the documentation required to store the media, and the copying and restoration process.

Calibrate and Maintain Process Monitoring and Data Collection

Testing and in-process monitoring of software should be performed according to an established QAP and procedures. Testing tools, system performance data and computer program performance measurement tools should be identified and qualified for their intended use. A tracking and reporting system on the status of the monitoring, control of software components should be established. Any modification to computer programs and related documentation should be reviewed before implementation by qualified and knowledgeable personnel.[90] Software embedded in monitoring and data collection equipment ("burned in" the integrated circuit) during manufacturing should have a version identifier. The version identifier should be noted and tracked as specified by the software configuration management work activity.

For additional guidance, refer to Section 5.2.6, *Software Design and Implementation,* Section 5.2.7, *Software Safety Analysis and Safety Design Methods* and Section 5.2.3, *Software Configuration Management*

[90] IAEA Technical Reports Series No. 397, *Quality Assurance for Software Important to Safety,* IAEA, 2000.

Criterion 6 – Performance/Design

(a) Design items and processes using sound engineering/scientific principles and appropriate standards.
(b) Incorporate applicable requirements and design bases in design work and design changes.
(c) Identify and control design interfaces.
(d) Verify or validate the adequacy of design products using individuals or groups other than those who performed the work.
(e) Verify or validate work before approval and implementation of the design.

The design of software should be defined, documented, controlled, verified and approved. Software should be designed using the most recent and sound coding principles and appropriate industry or government consensus standards. Applicable software requirements should be specified on a timely basis and translated into software design. Software design interfaces should be identified and controlled. The adequacy of software design and associated coding should be independently verified or validated by others who did not perform the design or the coding prior to its approval for use. The adequacy review should cover three phases of software development: requirements identification, design, and implementation.

Requirements Identification Phase

This phase consists of developing a software requirement specification (SRS) to defines and documents relevant functional, performance, safety, reliability and maintainability requirements. Design input and constraints, as well as interfaces and response to anticipated errors and failure modes, should also be defined in the SRS. These requirements should be traceable throughout the software development life-cycle.

Design Phase

The design phase consists of developing and documenting a detailed description of the computer program, the overall structure of the software, and the reduction of the overall structure into physical solutions. The same steps and processes for designing the software should apply regardless of the grade level or classification. However, the level of detail and required effort may vary greatly depending on the software classification category or grade level. The programming logic and data structures should be determined and clearly identified. The level of detail for documenting the design and the review of the documentation should be consistent with the risk associated with the software failure category or grading level.

Implementation Phase

The implementation phase consists of developing the source code (design output) using a standard programming language or other form suitable for compilation or translation into an executable code. The design, as described in the software design description (SDD), should be used as the basis for software development and should be modified to reflect modifications identified during the implementation phase.

For additional guidance, refer to Section 5.2.6, *Software Design and Implementation,* Section 5.2.8, *Verification and Validation*, Section 5.2.5 *Software Requirements Identification and Management* and Section 5.2.7, *Software Safety Analysis and Safety Design Methods.*

Criterion 7 – Performance/Procurement

(a) Procure items and services that meet established requirements and perform as specified.
(b) Evaluate and select prospective suppliers on the basis of specified criteria.
(c) Establish and implement processes to ensure that approved suppliers continue to provide acceptable items and services.

Software procurements may involve acquisition of software product or services. Software products may include a software package custom-developed to specific requirements of the procuring organization, or a software package previously developed (e.g., commercial off-the-shelf). Appropriate controls should be placed on the procurement process to ensure that it clearly states or references requirements and acceptance criteria. These should be based on a flow-down of the software design requirements for custom software.

Acquisition documents should specify applicable QA requirements for contracted software development and software services, and should identify all documentation, plans, and procedures to be supplied by the vendor. Vendor-supplied documentation should include, as a minimum:

- requirements documentation;
- design and implementation documentation;
- inspection and testing documentation;
- change documentation; and
- user documentation.

Refer to Section 5.2.4, *Procurement and Supplier Management, of this Guide for additional information.*

Criterion 8 – Performance/Inspection and Acceptance Testing

(a) Inspect and test specified items, services, and processes using established acceptance and performance criteria.
(b) Calibrate and maintain equipment used for inspections and tests.

The inspection and acceptance testing of software should include objective evidence of the review of software activities, life-cycle documentation, and test reports to ensure that the software:

- adequately and correctly performs all intended functions;
- properly handles abnormal conditions and events; and
- does not perform any unintended function that, either by itself or in combination with other functions, can degrade the intended output of the software.

The inspections may range from a group of subject matter experts (SME) conducting a formal document review meeting to a single SME performing a desktop check of the documentation or the code. Inspection and testing activities should be performed by a competent individual or group independent from the design and implementation group.

Configuration-controlled items should be under a configuration change control procedure prior to acceptance testing. Acceptance testing should be planned and performed for all documented software design requirements.

Test plans, test cases, and test results should be documented, reviewed, and approved prior to use of the software, and then placed under configuration change control. For lower-grade level software, the test plans, cases and results may be combined into a single document reviewed after completion of acceptance testing. The review report should be signed by the authors. Issues discovered during testing should be documented and dispositioned prior to approval of the test results. For software at high grade levels, each issue and its disposition should be documented.

Software modifications should be subject to selective regression testing to:

- Detect errors introduced during the modification of the systems or system components;
- Verify that the modifications have not caused unintended adverse effects; and
- Verify that the modified systems or system components still meet the specified requirements.

For additional guidance, refer to Section 5.2.8, *Verification and Validation*.

Criterion 9 – Assessment/Management Assessment

Ensure that managers assess their management processes and identify and correct problems that hinder the organization from achieving its objectives.

Management assessments should be performed periodically to ensure that the organization is meeting its mission objectives. Management decisions about how a software project is implemented can have significant effects on the safety and reliability of the software. Performing such assessments is one of many ways to identify strengths and improvement opportunities, and to also correct problems in the software development process.

The management assessment process is an integral part of the overall assessment process. Management personnel involved in the assessment process can provide the organization a more comprehensive perspective in understanding the broader implications of improvement opportunities and corresponding corrective actions. These assessments are self-initiated and self-conducted, thereby allowing the organization to identify problems and initiate corrections prior to being subjected to external review. Management assessments could focus on assessing the overall SQA program, the corrective action program as it relates to software or any other SQA element.

Management assessments of software are also critical, since software processes affect not only the software item they are associated with but also any other activities that rely on software. In addition, software processes crosscut other project objectives, such as schedule, cost, and resources, associated with a given project. Reviewing work associated with any software process is crucial and should be conducted early in the life-cycle of the software. This will ensure that any identified unaddressed problems are not compounded as the project moves ahead.

Criterion 10 – Assessment/Independent Assessment

(a) Plan and conduct independent assessments to measure item and service quality.
(b) Measure the adequacy of work performance and promote improvement.
(c) Establish sufficient authority and freedom from line management for independent assessment teams.
(d) Ensure persons who perform independent assessments are technically qualified and knowledgeable in the areas to be assessed.

Independent assessments can be internally or externally initiated, but they are conducted by a technically qualified independent organization not involved in the development of the software. Code development can often be considered a closed process. However, an external review can lead developers and other staff to see potential problems before the system is deployed

Independent assessments should address the design of a software-based process control system or a complex database application that affects the organizational mission, and other similarly-directed software systems.

Independent assessments are typically risk-informed. Thus, it may not be prudent to perform independent assessments of lower-grade software when such resources could be better applied to high-risk software projects. Moreover, high-risk software projects should be conducting assessments as they progress through the software acquisition or development life-cycle.

Examples of independent types of assessments include:

- Code walkthrough: provides assurance of the fitness for purpose of the algorithm or code and assesses the competence or output of an individual or team;
- Functional and physical configuration audit: a functional configuration audit ensures that functional and performance attributes of a configuration item are achieved, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation; and
- Peer review: review by others who have sufficient knowledge in the software. This may include but not limited to periodic and operational testing of software to confirm that the software elements are operating as required and provide assurance that the software meets the program's mission and quality objectives.

APPENDIX B. PROCEDURE FOR ADDING, REVISING OR DELETING SOFTWARE IN THE DOE SAFETY SOFTWARE CENTRAL REGISTRY

B.1 INTRODUCTION

B.1.1 PURPOSE

The Department of Energy (DOE) Safety Software Central Registry (CR) contains “toolbox codes.” These qualified codes are used to support safety analysis of DOE facilities. The purpose of this appendix is to outline a procedure for modifying the CR by adding new codes or updating existing codes. The procedure ensures compliance with the software quality assurance (SQA) requirements of DOE O 414.1D, *Quality Assurance* by providing suitable criteria. Information is also presented on procedures to (1) revise or update a toolbox code and (2) remove a code from the CR.

The CR can be accessed at http://energy.gov/ehss/safety-software-quality-assurance-central-registry

B.1.2 SCOPE

The scope of this procedure includes any software application used by DOE or its contractors for inclusion in the CR.

B.1.3 FUNCTIONS

Procedures for adding, revising and removing software in the CR are as follows:

Code Sponsor: The code sponsor can be the code’s developer or a user organization. In either case, this party is responsible for documenting the complete SQA program for the code. The code sponsor is also responsible for documenting the rationale for listing the subject code in the CR.

SQA Evaluator: An SQA evaluator should have had no involvement in the development of the code. Review organizations or individuals should have (a) a thorough understanding of applicable SQA requirements, (b) expert level knowledge and application experience with the code in question, and (c) an awareness of the overall context for the use of the subject code in safety analysis. The SQA evaluator is responsible for documenting the evaluation of the candidate code, and based on this evaluation, may or may not conclude that the SQA of the code satisfactorily meets requirements for listing in the CR.

CR Listing Approver: DOE's Office of Quality Assurance (AU-33) in the Office of Nuclear Safety (AU-30) reviews and evaluates SQA of the candidate code and makes recommendations to the Director, AU-30, as to whether the candidate code should be listed in the CR.

Before a formal request for listing a code in the CR, the code sponsor should contact DOE's Office of Quality Assurance to review documentation that will have to be submitted in support of the request.

B.2 PROCESS

B.2.1 ADDING SOFTWARE APPLICATIONS TO THE CENTRAL REGISTRY

In the application, the code sponsor should justify why the code should be listed in the CR. The justification statement should establish for the candidate code:

- Widespread current use in the DOE complex,
- Potential for other safety-related applications in the DOE complex,
- Documentation is adequate to meet DOE's SQA requirements, and
- A demonstrated and quantifiable benefit for listing the code in the CR.

The code sponsor may be requested to provide information on the programs and procedures associated with the development, maintenance, and use of the subject code. The code developer is expected to provide necessary SQA documentation to address the work activities in DOE O 414.1D, Attachment 4.

Formal documentation to demonstrate that the SQA work activities have been performed is preferred. However, in cases where the software was not developed using formal SQA procedures, other less formal forms of documentation such as files, reports, and meeting notes can provide in the aggregate the desired confirmation.

B.2.1.1 Evaluation Process

The SQA evaluator performs and documents a review of the code, using inputs from the code developer. In cases where the code developer is unable to supply requested inputs, the SQA evaluator may consider alternative sources of information such as previous reviews,[91] older documentation from the code developer, technical and journal articles, and previous software comparison studies.

[91] If previous reviews are used in whole or in part, the code developer should confirm that the older review results are still applicable.

Table B-1, Software Quality Assurance Work Activities and Corresponding Documentation for Demonstrating Compliance

DOE O 414.1D SQA Work Activity	SQA Documents
1. Software Project Management and Quality Planning	– Software Project Management Plan (SPMP) and/or – Software Quality Assurance Plan (SQAP) – Software Safety Plan
2. Software Risk Management	– Various document types can be used to cover risk management
3. Software Configuration Management	– Software Configuration Management Plan (SCMP) or related documents
4. Procurement and Supplier Management	– Contractual documents or other procurement and use agreement documentation
5. Software Requirements Identification and Management	– Software Requirements Specifications (SRS) or related document
6. Software Design and Implementation	– Software Design Description (SDD), Model Description, Programmer’s Reference Manual, or other related documents
7. Software Safety	– SDD – Software Safety Analysis documentation
8. Verification and Validation	– Verification and Validation Report – Test Case Description and Outcome Report; Other testing documents
9. Problem Reporting and Corrective Action	– Software Error Notification and Corrective Action Report
10. Training of Personnel in the Design, Development, Use and Evaluation of Safety Software	– User Instructions or User Manuals – Training Packages and User Qualification
11. Model Validation and Evaluation	– Test results and evidence that code output was compared to experimental results or against equivalent output from an independent code and differences resolved

The size of the SQA evaluation effort, whether put forth by one individual or by a team of subject matter experts, depends on the complexity of the code application. Evaluation of the SQA work activities covered in Table B-1 above should use a sub-matrix of finer criteria to adequately evaluate the constituent parts of the requirement. A qualitative ranking of compliance items, which has been used in the evaluation of current toolbox codes, consists of applying four terms to define compliance with SQA requirements. These terms are: *Yes (meets requirement), Partial (some but not all criteria are met), No (does not meet requirement),* and *N/A (requirement is not applicable)*. Upon completion of the evaluation of each of the SQA work activities, the SQA evaluator can review results as a whole and render an overall assessment. The process should lead to documenting findings in a verifiable and objective manner.

The overall evaluation process is shown schematically in Figure B-1.Table B-2 provides guidance on evaluating toolbox-equivalent candidate codes, defined as custom-developed software.

While grading a code as Level C is possible, most candidate codes for the CR will be categorized as Level A or B.

The SQA evaluations performed on the six initial toolbox codes are a reasonable approximation of the level of detail desired for SQA evaluation of a new candidate code.
SQA evaluation criteria for adding codes to the CR can be downloaded from the CR website: (http://energy.gov/ehss/safety-software-quality-assurance-central-registry).

Table B-2. Plan for Evaluation of Candidate Codes for the Central Registry

Step	Procedure
1. Review Documentation	• Determine that sufficient information is provided by the software developer to allow proper classification of the software; • Review developer reports, previous evaluations, and conference and journal submittals, etc.; and • Interview code developer.
2. Evaluate Justification (Rationale) for Including Software in CR	• Review code sponsor's documents to determine: • Widespread use or prospect for significant use of the code across the DOE complex for safety related applications; • Methods to ensure proper SQA related information, error reporting, configuration control and other SQA management interfaces with the CR; and • Demonstrated and quantifiable benefit for designating the software for the CR.
3. Assess Software Project Management and Software Quality Assurance Plans	• Review SPMP and SQAP for: • required activities, documents, and deliverables and • level and extent of reviews and approvals, including internal and independent review. • Confirm that actions and deliverables (as specified in the SQAP) have been completed and are adequate. • Review engineering documentation identified in the SPMP and SQAP, including: (1) software risk management documents; (2) software configuration management plan; (3) procurement and supplier management documents; (4) software requirements specifications; (5) software design, model description, programmer's reference, and related documents; (6) software design and related documents; (7) verification and validation, test report, and other documents;

Step	Procedure
	(8) software error notification and corrective action reports; (9) user instructions, user manuals, and training packages/user qualification documents; and (10) model validation and evaluation.
4. Assess SQA Work Activities	• Review SQA documentation against detailed criteria found in DOE O 414.1D SQA Work Activities: • software project management & quality planning; • software risk management; • software configuration management; • procurement and supplier management; • software requirements identification and management; • software design and implementation; • software safety; • verification and validation; • problem reporting and corrective action; • training of personnel in the design, development, use and evaluation of safety software; and • model validation and evaluation.
5. Document Evaluation of the code	• Use past evaluation reports as template.

B.2.1.2 Submittal to the Central Registry

Once the SQA evaluation has been conducted and documented, the code may be submitted to the CR as one of the following cases:

Case 1: The evaluation report indicates that software meets all requisite criteria in the SQA work activities, and the report makes no recommendations about corrective actions.

Case 2: The evaluation report has identified one or more criteria not met for the subject code; the code sponsor has made improvements acceptable to the evaluator.

However, the code sponsor can document a compelling technical basis for submitting the code as "toolbox-equivalent" to the CR despite a failure to meet SQA criteria. The technical basis should include a code guidance report that points out specific limitations and weaknesses and provides instructions to the user on informed use of the subject code despite identified flaws and vulnerabilities.

Guidance reports prepared for the initial six codes designated for the CR may be downloaded from the DOE SQA website at http://energy.gov/ehss/safety-software-quality-assurance-central-registry.

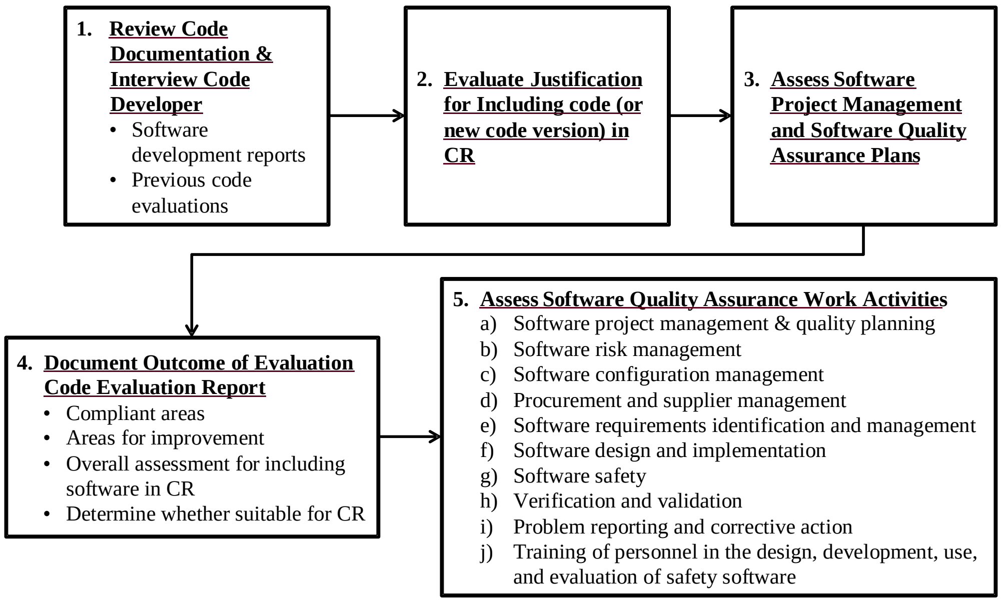
1. Review Code Documentation & Interview Code Developer
• Software development reports
• Previous code evaluations
2. Evaluate Justification for Including code (or new code version) in CR
3. Assess Software Project Management and Software Quality Assurance Plans
4. Document Outcome of Evaluation Code Evaluation Report
• Compliant areas
• Areas for improvement
• Overall assessment for including software in CR
• Determine whether suitable for CR
5. Assess Software Quality Assurance Work Activities
a) Software project management & quality planning
b) Software risk management
c) Software configuration management
d) Procurement and supplier management
e) Software requirements identification and management
f) Software design and implementation
g) Software safety
h) Verification and validation
i) Problem reporting and corrective action
j) Training of personnel in the design, development, use, and evaluation of safety software

Figure B-1, Flow Sheet for Software Evaluation

If all substantive issues in either Case 1 or Case 2 are satisfactorily dispositioned, the code sponsor may move forward the candidate toolbox code documentation along with the request to review the evaluation as part of the toolbox code application.

As the CR Listing Approver, the DOE Office of Quality Assurance will review the submittal. Table B-3 lists several of the key acceptance criteria for rendering a decision to list the candidate code in the CR. A decision on designation of the candidate code as a toolbox code application will be communicated to the code developer and evaluator organizations. If the decision is favorable, the appropriate links will be provided for the code in question, and a general notice will be posted on the CR Web site. Additional notification methods may be implemented to ensure broad notification of the changes in the codes listed in the CR.

If, on the other hand, issues with the subject code are irreconcilable, then the code sponsor is advised not to proceed further with the submittal process. It may be prudent to examine continued use of the code at the site in question, and explore use of an alternative code, such as codes currently contained in the CR, for the specific safety application.

Table B-3, Primary Criteria for Deciding on Inclusion of a Code to the Central Registry

Phase	Criterion
1. Rationale for Adding a Code to CR	a. Widespread use of the code across DOE complex for safety related applications. b. Methods to ensure proper code information, error reporting, configuration control, and other SQA management interfaces with the CR. c. Demonstrated and quantifiable benefit for designating the code as a CR toolbox code.
2. SQA Technical Basis	a. The SQA evaluation document adequately demonstrates that the candidate code has met all major requisite criteria, and no criterion is evaluated as "No (= not met)." If remedial tasks were cited before all criteria are considered met, it is determined that these have been completed. or b. The SQA evaluation document has identified one or more criteria not compliant for the subject code based on the gap analysis. However, a compelling technical basis is made for submitting the code as "toolbox-equivalent" to the CR. Part of the technical basis should include a guidance report that points out specific limitations, weaknesses, and provides instructions to the user on informed use of the subject code despite identified gaps and other vulnerabilities.

B.2.2 REVISIONS TO CODES LISTED IN THE CENTRAL REGISTRY

In the life-cycle processes associated with most code applications, updates, improvements, and modifications will be made. A revised code such as a new version may also be submitted for inclusion in the CR, generally accompanied by removal of the older version.

The same process is followed for a revised code to be placed in the CR as is outlined above for new code applications. The steps may be summarized as follows.

1. The code sponsor identifies the SQA evaluator organization.
2. The evaluator performs a complete evaluation of all aspects of the new code version, emphasizing new and revised aspects of the code application.
3. Upon conclusion of the evaluation and issuance of the SQA evaluation, if the code sponsor decides that the code meets all requisite criteria for the ten SQA work activities plus model validation/performance, the revised code may be submitted to the CR.
4. The code sponsor requests a review of the evaluation and designation of the code as a toolbox code application. All supporting documentation should be transmitted as attachments to the request.
5. DOE's Office of Quality Assurance will review the submittal and make a decision. If the decision is positive, the appropriate links will be provided for the code version, and a general notice will be posted on the CR website regarding a new code revision. In

parallel with this action, the older code version will be removed from the CR and designated as an “archived toolbox version.”

B.2.3 REMOVAL OF CODES FROM THE CENTRAL REGISTRY

Codes are also subject to being removed from the CR. Causes for removal include:

1. The code developer indicates that older versions will no longer be supported and elects to retire the code.
2. New survey information indicates that few if any sites are using the code and other codes are being used for the specified safety applications.
3. DOE’s Office of Quality Assurance decides to remove the code based on evidence of vulnerabilities or unsatisfactory code performance. Significant software errors in the subject code may lead to this outcome.

A comment period of 60 days is allowed after initial notification that a code will be removed from the CR. Removal may proceed provided no compelling arguments are received during the comment period that the code should be retained in the CR. The initial and final notification should explain the basis for the removal and provide supporting documentation.

Upon removal from the CR, the code is designated an “archived code.”

B.2.4 ISSUE RESOLUTION AND ACTION COMMUNICATION

Actions regarding the CR will be communicated to DOE staff, DOE code users, and stakeholder groups. The level of detail in the communication is within the discretion of DOE’s Office of Quality Assurance.

Announcements of CR actions may be posted on the CR website, communicated directly to Program Secretarial Offices, or both. The URL for the CR website is:

http://energy.gov/ehss/safety-software-quality-assurance-central-registry

APPENDIX C. SELECTING AND IMPLEMENTING APPLICABLE WORK ACTIVITIES

C.1 INTRODUCTION

This appendix provides additional detail and examples on how to implement software quality assurance (SQA) work activities listed in Attachment 4 of DOEO 414.1D, *Quality Assurance* and further described in Section 5.2 of this Guide. These work activities depict a systematic application of the software development life-cycle processes for both safety software and OSW. This appendix uses the six-step approach as previously shown in Figure 5.1-1.

C.2 APPROACH

C.2.1 *Step 1: Identify Applicable Consensus Standards*

As part of a quality assurance program (QAP), DOE O 414.1D requires the selection, of appropriate national or international consensus standards, consistent with contractual and regulatory requirements and Secretarial Officer direction. Select and document the appropriate national or international consensus standards and document the selection in the site-wide QAP.

Standards for Safety Software

ASME NQA-1-2008/ NQA-1a 2009 is used to satisfy DOE O 414.1D requirements for safety software. However, DOE O 414.1D and later editions of ASME NQA-1 permit use of other national or international consensus standards that provide an equivalent level of QA.

Standards for OSW

DOE O 414.1D requires that all software meet the applicable QA requirements in Attachment 2 using a graded approach. DOE O 414.1D also requires the use of appropriate national or international consensus standards, in whole or in part, consistent with regulatory requirements and Secretarial Officer direction, to clearly identify which standards, or parts of the standards, are used in the QAP.

C.2.2 *Step 2: Determine Software Category, Sub-category and Grading Level*

SQA is applied based on three factors:

- Software category;
- Software sub-category; and,
- Software grading level.

As discussed in Section 2.2 and shown in Table C-1, this Guide identifies two software categories and six software sub-categories.

Table C-1 Software Categories and Subcategories

Software Category	Software Subcategory
1. **Safety Software**	Safety System Software (SSS)
	Safety and Hazard Analysis and Design Software (SHADS)
	Safety Management and Administrative Control Software (SMACS)
2. **OSW**	Safety Affecting Software (SAS)
	Critical Software (CS)
	General Software (GS)

Safety Software Categories

For safety software, determine the software category and sub-category using the definitions for in DOE O 414.1D and in Section 2.2.1 of this Guide.

OSW Categories

For OSW, determine the software category and subcategory using the governing QAP and Section 2.2.2 of this Guide.

It is important to have a clear understanding of the software category, as this determination involves software function and applicability, and is therefore *application-specific*. The same software could be determined to be safety software in one application for example, in a nuclear facility, and determined to be OSW in a non-nuclear facility application. To facilitate making the correct determination, involve personnel qualified and knowledgeable of the software application such as design authority, system engineers, software owners, software users, etc. Procedures including forms may be developed to document the process and supporting rationale. If application of the software significantly changes, review the original determination and revise it, as necessary.

Grading Levels

Section 2.2 of this Guide offers guidance on how to determine the appropriate grading level, based on software risk and consequences of software failure.

Safety Software:

DOE O 414.1D requires that safety software grading levels are established, documented, and approved by the responsible DOE authority for each site. Such grading levels are then used in conjunction with the software category and sub-category to apply the appropriate level of SQA. The grading level examples are shown in Table C-2. Use the grading level designations (i.e., A, B, C), with the A grading level having the highest consequence of failure and the C grading level having the lowest consequence of failure, as described in Table 2.2-1.

OSW:

DOE O 414.1D requires that all software, therefore including OSW, meet applicable QA requirements using a graded approach. In applying a graded approach, the grading levels shown in Table C-2 may be used for OSW. Grading level designations range from 1 to 4, with grading level 1 having the lowest consequence of failure and grading level 4 having the highest consequence of failure, as described in Table 2.2-2.

Table C-2 Example Software Category, Subcategory and Grading Level

Software Category	**Software Subcategory**	**Grading Level**
1. Safety Software	Safety System Software (SSS)	A, B, C
	Safety and Hazard Analysis and Design Software (SHADS)	A, B, C
	Safety Management and Administrative Control Software (SMACS)	A, B, C
2. OSW	Safety Affecting Software (SAS)	4, 3
	Critical Software (CS)	3, 2
	General Software (GS)	1

C.2.3 *Step 3: Determine Applicability of the Work Activity*

Applicability is a concept sometimes confused with grading. Applicability is based on the scope and control of the software project, not on the software category, sub-category or grading level. It means to apply work activities only if they are:

- Within the scope of the software project (germane to the software/work at hand); and,
- Within the control of the project.

For example, if the software project involves purchasing software and no software design is planned by or within the control of procurement, then the work activity entitled “Software Design and Implementation” does not apply to the purchaser’s project. In this example, the purchaser may note “not applicable” in the software planning documentation for the software design and implementation work activity. Applicability of work activities is illustrated in Example A below.

Example A: Applicability of Work Activities

A DOE facility is buying the same process piping design and analysis software for two facilities, one a nuclear facility and the other an administration building.

For the nuclear facility, the facility determined the software category as SSS, the sub-category as SHADS, and the grading level as B.

For the non-nuclear facility, the facility determined the software category as OSW, the sub-category as GS, and grading the level as 1.

Although the software for the facilities differs in category, sub-category and grading level, the software scope of the project in both cases is to purchase COTS from a supplier. The project does not involve any software design or development by the facility. Accordingly, the design and implementation work activity is not applicable.

C.2.4 *Step 4: Define the Work Activity with Consensus Standards*

Use the site-adopted consensus standards to determine and document necessary work activities.

See Example B for safety software application.

Example B: Safety Software - Define the Work Activity with Consensus Standards

Table C-3 shows how a DOE facility defined the requirements for the working activity *problem reporting and corrective action*. The facility uses ASME NQA-1-2008/ NQA-1a 2009 as its consensus standard; the work activity satisfies one cited section of DOE O 414.1D and two sections from ASME NQA-1-2008/ NQA-1a 2009.

Table C-3 Safety Software - Problem Reporting and Corrective Action (Example B)

Source of Requirements	Requirements Definition
DOE O 414.1D, Att. 2, § 3 – Criterion 3 – Management/Quality Improvement	a. Establish and implement processes to detect and prevent quality problems. b. Identify, control, and correct items, services, and processes that do not meet established requirements. c. Identify the causes of problems, and include prevention of recurrence as a part of corrective action planning. d. Review item characteristics, process implementation, and other quality related information to identify items, services and processes needing improvement.
ASME NQA-1, Part I, Requirement 16 – Corrective Action	Conditions adverse to quality should be identified promptly and corrected as soon as practicable. In the case of a significant condition adverse to quality, the cause of the condition should be determined and corrective action taken to preclude recurrence. The identification, cause, and corrective action for significant conditions adverse to quality should be documented and reported to appropriate levels of management. Completion of corrective actions should be verified.
ASME NQA-1, Part II, SP 2.7; § 204 – Problem Reporting & Corrective	*(a)* Method(s) for documenting, evaluating, and correcting software problems should *(1)* Describe the evaluation process for determining whether a reported problem is an error or other type of problem (e.g., user mistake).

Source of Requirements	Requirements Definition
Action	*(2)* Define the responsibilities for disposition of the problem reports, including notification to the originator of the results of the evaluation. *(b)* When the problem is determined to be an error, the method should provide, as appropriate, for: *(1)* How the error relates to appropriate software engineering elements; *(2)* How the error impacts past and present use of the computer program; *(3)* How the corrective action impacts previous development activities; and, *(4)* How the users are notified of the identified error, its impact; and how to avoid the error, pending implementation of corrective actions. The problem reporting and corrective action process should address the appropriate requirements of Part I, Requirement 16.

C.2.5 *Step 5: Grade by Specifying Appropriate Work Activity Requirements*

Grade the work activity by specifying the applicable elements of the consensus standard for each software category, sub-category, and grading level. This specification is typically documented in site SQA program documents.

For safety software, and for OSW with high to very-high consequence of software failure, all applicable consensus standard requirements should be applied. For software where there is a lower consequence of failure, a graded approach should be applied commensurate with the risk.

Example C below addresses safety software, while Example D addresses OSW.

Example C: Safety Software - Grading by Specifying the Appropriate Elements of Consensus Standard(s) for a Work Activity

A DOE nuclear facility uses safety software that spans three safety software categories and grading levels A, B, and C. The facility uses NQA-1 as its consensus standard in its approved QAP for these facilities. Three situations are dealt with below.

1. SSS in a PLC is used to maintain the relative pressure in a safety class glovebox containing radioactive materials. The consequences of software are high and hence grading level A has been assigned.

2. SHADS is used by safety basis personnel to model postulated releases of radioactive material. The modeling software provides dispersion information that the site uses to determine unmitigated accident consequences. After application of selected controls, the same software is used for determining mitigated consequences. The consequence of failure of this software has been designated to be moderate suggesting grading level B.

3. SMACS is used to control the material at risk for conformance with a nuclear facility safety basis. The consequence of failure of this software has been determined to be relatively low because other available hazard controls can compensate for the failure; hence, grading level C is assigned.

The facility has determined that SQA work activity *Problem Reporting and Corrective Action* applies to each of these three software applications. Using the requirements in Table C-3 and considering the consequences of software failure, the facility has arrived at the conclusions displayed in Table C-4.

Due to the potential high consequences of software failure, more requirements of the consensus standard are specified for safety software to reduce the risk associated with safety software. Often, for safety software, much of the software grading is performed in the rigor used to satisfy this requirement (See Step 6, Grade by Applying Appropriate Rigor).

Table C-4 Safety Software: Grading of Problem Reporting and Corrective Action (Example C)

Source of Requirement	**Grading Work Activity**		
	SSS Safety Class Glovebox Grading Level A	**SHADS Radionuclide Dispersion Modeling, Grading Level B**	**SMACS MAR Control, Grading Level C**
DOE O414.1D, Att. 2, S 3 – Criterion 3 – Management/Quality Improvement	Full	Full	Full
ASME NQA-1, Part I, Requirement 16 – Corrective Action	Full	Full	Full
ASME NQA-1, Part II, SP 2.7; S 204 – Problem Reporting & Corrective Action	Full	Full	Partial
Full = The requirement is used in its entirety. Partial = Selected elements are not used.			

Example D: OSW - Grading by Specifying the Appropriate Elements of Consensus Standards for a Work Activity

A DOE facility uses OSW in three applications. Grading levels for OSW are described in the facility's QAP. Three situations are dealt with below.

SAS is used to control negative ventilation pressure in gloveboxes containing dangerous biological agents and toxins. The consequence of software failure has been determined to be relatively high and hence grading level 4 has been assigned.

CS is used by emergency response personnel in accordance with the site's Continuity of Operations Plan. This modeling software provides biological agent or toxin dispersion information to guide emergency response actions in the event of a containment breach. The consequence of failure of this software has been determined to be moderate, suggesting grading level 3 is appropriate.

GS is used to control heating, ventilation and cooling for personnel comfort in an office building. The consequence of failure of this software has been determined to be relatively low and accordingly, grading level 1 has been assigned.

The facility has determined that SQA work activity *Problem Reporting and Corrective Action* applies to each of these software applications. The facility uses ANSI/ISO/ASQ (E) Q9001-2008, *American National Standard Quality Management Systems – Requirements*, hereafter referred to as ISO 9001, as its QAP consensus standard.

The facility has identified the requirements for problem reporting and corrective actions as shown in Table C-5.

Table C-5 OSW: Definition of Problem Reporting and Corrective Action (Example D)

Source of Requirements	Requirements Definition
DOE O 414.1D, Att. 2, § 3 – Criterion 3 – Management/Quality Improvement	a. Establish and implement processes to detect and prevent quality problems. b. Identify, control, and correct items, services, and processes that do not meet established requirements. c. Identify the causes of problems, and include prevention of recurrence as a part of corrective action planning. d. Review item characteristics, process implementation, and other quality related information to identify items, services and processes needing improvement.
ISO 9001, § 8.5.2, Corrective Action	The organization should take action to eliminate the causes of nonconformities in order to prevent recurrence. Corrective actions should be appropriate to the effects of the nonconformities encountered. A documented procedure should be established to define requirements for: a. Reviewing nonconformities (including customer complaints). b. Determining the causes of non-conformities. c. Evaluating the need for action to ensure that nonconformities do not recur. d. Determining and implementing action needed. e. Records of the results of action taken (see [ISO 9001:2008 Section] 4.2.4). f. Reviewing the effectiveness of the corrective action taken.

Source of Requirements	Requirements Definition
ISO 9001, § 8.5.3, Preventive Action	The organization should determine action to eliminate the causes of potential in order to prevent their occurrence. Preventive actions should be appropriate to the effects of the potential problems. A documented procedure should be established to define requirements for: a. Determining potential nonconformities and their causes. b. Evaluating the need for action to prevent occurrence of non-conformities. c. Determining and implementing action needed. d. Records of results of action taken (see [ISO 9001:2008 Section] 4.2.4). e. Reviewing the effectiveness of the preventive action taken.

The facility has specified the requirements for problem reporting and corrective action as shown below in Table C-6.

Table C-6 OSW: Grading of Problem Reporting and Corrective Action (Example D)

Source of Requirements	**OSW Grading of Work Activity**		
	Safety Affecting Software (SAS) Biological Hazard Glovebox Ventilation Grading Level 4	**Critical Software (CS) Modeling of Biological Hazard Release Grading Level 3**	**General Software (GS) Office Building Comfort Heating/Cooling Grading Level 1**
DOE O 414.1D, Att. 2, § 3 – Criterion 3 – Management/Quality Improvement	Full	Full	Full
ISO 9001, § 8.5.2, Corrective Action	Full	Full	Partial
ISO 9001, § 8.5.3, Preventive Action	Full	Partial	None
Full = The requirement is used in its entirety. Partial = Selected elements are not used.			

C.2.6 *Step 6: Grade by Applying Appropriate Rigor*

Examples C and D show how to grade by specifying the appropriate elements of consensus standard(s) for a work activity. This section shows how to grade by applying appropriate rigor.

Rigor, as defined in this Guide, is the level of detail and thoroughness in the analysis, documentation and action used to satisfy a requirement. The level of rigor is an important tool for applying the graded approach to risk level. Example E illustrates how different levels of rigor may be may be applied to a work activity for OSW subcategories. Examples E shows how applicable requirements were satisfied in each scenario employing different levels of rigor.

Example E: OSW – Project Management and Quality Planning

As stated in Example D, this DOE facility uses three subcategories of OSW. Three situations are dealt with below.

SAS is used to control negative ventilation pressure in gloveboxes where dangerous biological agents and toxins are confined. The consequence of software failure has been determined to be relatively high and hence grading level 4 has been assigned. CS is used by emergency response personnel in accordance with the site's Continuity of Operations Plan. This modeling software provides biological agent or toxin dispersion information to guide emergency response actions in the event of a containment breach. The consequence of failure of this software has been determined to be relatively moderate, suggesting grading level 3 is appropriate. GS is used to control heating and cooling ventilation for personnel comfort in an office building. The consequence of failure of this software has been determined to be relatively low and accordingly, grading level 1 has been assigned.

Grade by Specifying Appropriate Rigor

The DOE facility discussed above developed three separate project plans, one for each software application, and based each plan on IEEE/ISO/IEC 16326-2009, *Systems and Software Engineering--Life Cycle Processes--Project Management.*

The plan for the SAS contained more detail than both the CS and GS software plans. The project team was larger and even more diverse than that for the CS and GS and accordingly required significant documentation of roles responsibilities and interfaces. The funding for the software project was only available in the current fiscal year. If the project wasn't completed by the end of the fiscal year, then it would be placed on hold until funding to complete it could be obtained. Accordingly, the project plan included several project risk management controls to closely monitor the project and promote prevention/early detection of delays and timely corrective action. The plan required an in-depth hazard analysis based on a nationally recognized standard[92] ; the plan required that multiple failures/common cause failures would be addressed as part of the hazard analysis. The plan required alternate calculations to check the software design.

The plan for the CS was lengthier as it had a much larger and diverse team and the software project was more complex than the GS software which required a hazard analysis using a "what if" technique. The plan specified that once the design was proven with both alternate calculations, in-depth test planning and testing of multiple test cases which bounded the possible applications would be performed. The plan required more documentation of roles, responsibilities, interface controls, etc. and alternate modeling calculations to prove the software design was acceptable.

[92] ANSI/ISA 84.00-01, *Application of Safety Instrumented Systems for the Process Industries*.

The plan for the GS was a few pages in length, did not require a hazard analysis or alternate calculations and emphasized the testing of the software. There was sufficient time to complete the project within the allocated annual budget without imparting operational risk to the DOE facility; accordingly very few project risk management controls were planned.

Example F: Applying Rigor in Acceptance Testing

For safety software used safety class glovebox ventilation (SSS), the facility required a stress test. The final acceptance test was run multiple times to ensure satisfactory performance prior to introduction of radioactive material into the glovebox.

For the OSW used for the office building's heating and cooling (GS), the facility did not require stress testing and required only one final acceptance test prior to building occupancy.

Example G: Applying Rigor in Software Design Change Reviews

For the OSW used to model biological agent and toxin releases for emergency response (CS), the facility required that a group of independent subject matter experts in various disciplines review software design changes.

By contrast, in the office building case, the facility required a less rigorous review of design changes.

APPENDIX D. QUALITY ASSURANCE STANDARDS FOR SOFTWARE

D.1 INTRODUCTION

. This Appendix reviews major national and international consensus standards and other guidance related to quality the application of software quality assurance (SQA) for their suitability to satisfy Department of Energy (DOE) Order (O) 414.1D, *Quality Assurance* requirements in whole or in part.

D.2 QUALITY ASSURANCE PROGRAM

DOE 414.1D also requires that appropriate national or international consensus standards be used to develop and implement, using a graded approach, quality assurance programs (QAPs) at DOE sites consistent with contractual and regulatory requirements. Consensus standards should be used for applying quality assurance (QA) to software activities.

D.3 QA PROGRAM STANDARDS VERSUS SOFTWARE STANDARDS

Numerous consensus standards have been developed that address every aspect of software. Other documents such as technical reports, agency directives, and industry guides may be useful for application of SQA, though they may not have been developed through an accredited consensus standards process. Because these standards could be interpreted as "QA standards," it is necessary to limit discussion of consensus standards to those that directly support compliance with DOE O 414.1D and the development of a contractor QAP that includes safety software and other software (OSW).

D.4 USE OF STANDARDS IN A QA PROGRAM

The majority of software consensus standards have been developed to address specific phases of software development or a single criterion within the QAP. Where this type of standard is used, it should be in the context of the broader QAP that includes all criteria necessary for effective QA. This approach will differentiate between QAP standards and standards that address a specific activity or criterion.

D.5 QA PROGRAM AND SOFTWARE QUALITY STANDARD REQUIREMENTS

Selection of QAP consensus standards for safety software should be based on:

- compatibility with DOE O 414.1D;
- relevance to facility or personnel safety;
- effect on the environment and programmatic mission;
- compatibility with a software consensus standard;
- applicability to software developed in-house, purchased, or modified;
- applicability to the entire software life-cycle; and

- application of commercial SQA practices.

D.6 NATIONAL STANDARD FOR NUCLEAR FACILITY QUALITY AND SOFTWARE

American Society of Mechanical Engineers (ASME) NQA-1-2008/NQA-1a-2009, *Quality Assurance Program for Nuclear Facility Applications*, includes SQA requirements that can be integrated applied directly to software. ASME NQA-1 2008/NQA-1a-2009 provides requirements for software quality in the context of an overall QA program. These sections are especially relevant to SQA:

- ASME NQA-1-2008/NQA-1a-2009, Part I, Requirement 3, Section 800, *Design Control*;
- Part II, Subpart 2.7, *Quality Assurance Requirements for Computer Software for Nuclear Facility Applications*; and
- Part IV, Subpart 4.1, *Guide on Quality Assurance Requirements for Software*.

ASME NQA-1-2008/NQA-1a-2009 is a practical choice for implementing DOE O 414.1D as it applies to safety software because it:

- is easily supplemented with standards promulgated by the International Atomic Energy Agency (IAEA), the International Electrotechnical Commission (IEC), and the Institute of Electrical and Electronics Engineers (IEEE);
- provides independence for development and verification;
- supports graded implementation; and
- is widely used among DOE's contractor QA programs.

ASME NQA-1-2012 provides additional standards guidance as noted below:

- Subpart 4.1.1, provides guidance on modifications of an ISO 9001:2008, *Quality Management Systems Standard*, for compliance with ASME NQA-1 2008.
- Subpart 4.1.2 describes how ASME NQA-1 2008 aligns with DOE's QA criteria.
- Subpart 4.1.4, provides guidance on modifications of an IAEA GS-R-3 Quality Program to meet ASME NQA-1 2008.
- Subpart 4.2.1, provides guidance on the graded application of nuclear quality assurance standards for research and development.
- Subpart 4.2.4, provides guidance on the *Control of Scientific Investigations.*

Institute of Electrical and Electronics Engineers (IEEE) Std 7-4.3.2-2003[93] describes an integrated approach to computer-specific requirements for developmental firmware, software, and hardware. This standard recommends a minimum set of functional and design requirements for computer components of a safety system employed in nuclear power generating stations.

[93] IEEE Std 7-4.3.2-2003, *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations,* Institute of Electrical and Electronic Engineers, 2003.

IEEE Std 1228[94] provides requirements for the development of a management plan and performance of safety software activities.

American National Standards Institute (ANSI)/American Nuclear Society (ANS)-10.4-2008[95] and ANSI/ANS-10.7-2013[96] supplements IEEE Std 7-4.3.2-2003 by targeting activities to improve the reliability of scientific and engineering computer applications while mitigating the risk of incorrect applications. IEEE has a complete suite of software and systems engineering standards applicable to safety software.

D.7 INTERNATIONAL CONSENSUS STANDARDS FOR QUALITY AND SOFTWARE

D.7.1 INTERNATIONAL ATOMIC ENERGY AGENCY (IAEA)

The IAEA develops international standards for nuclear safety. The IAEA offers standards, guides, and requirements for all aspects of nuclear facility safety including software. The requirements and guidance for nuclear facility quality are addressed in IAEA Safety Series No. 50-C/SG-Q, *Quality Assurance for Safety in Nuclear Power Plants and other Nuclear Installations*, and Safety Guides 50-SG-Q1 through Q14, respectively. The IAEA code quality requirements closely parallel the DOE O 414.1D.

IAEA safety software guidance is detailed in Technical Reports Series No. 397, *Quality Assurance for Software Important to Safety*. This series provides information and guidance on QA programs, covering the entire life-cycle of software important to safety. Technical Report 397 (2000) was offers implementation guidance tied to the QA program requirements found in the IAEA code.

The IAEA Safety Guide Series No. NS-G-1.1, *Software for Computer Based Systems Important to Safety in Nuclear Power Plants*, provides expanded information that can be fully integrated with the ASME NQA-1 2008/NQA-1a-2009 and DOE O 414.1D to produce an effective quality program for software.

D.7.2 INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC)

The IEC is responsible for several software standards. These standards, referenced in the IAEA's Technical Report 397 are:

[94] IEEE Std 1228- 1994, Standard for Software Safety, IEEE, 1994.
[95] American National Standards Institute (ANSI)/American Nuclear Society (ANS) ANSI/ANS-10.4-2008: *Verification and Validation of Non-Safety-Related Scientific and Engineering Computer Programs for the Nuclear Industry.*
[96] ANSI/ANS 10.7-2013, Non-Real-Time, High-Integrity Software for the Nuclear Industry—Developer Requirements, ANSI/ANS, March 2013

- IEC-60880 edition 2, *Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions*,
- IEC 60987 edition 2.*1, Nuclear power plants - Instrumentation and control important to safety - Hardware design requirements for computer-based systems,*
- IEC 61226 edition 3.0, *Nuclear Power Plants—Instrumentation and Control Systems Important for Safety—classification*,
- IEC 61508 edition 2 Parts 1-7, *Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems*, and
- IEC 61511 Parts 1-3, *Functional Safety – Safety Instrumented Systems for the Process Industry Sector*.

D.7.3 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO)

ISO is responsible for ISO 9001-2008, *Quality Management Systems – Requirements.* ASME NQA-1-2012, Subpart 4.1.1, provides guidance on modifications of an ISO 9001:2008, *Quality Management Systems Standard*, for compliance with ASME NQA-1 2008. ISO 9001 does not specifically address computer software. However, ISO/IEC has issued technical reports related to system and software engineering. They are found on the following website: http://www.iso.org/iso/home/search.htm?qt=software+quality+assurance&sort=rel&type=simple&published=on

D.8. NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA)

NASA's SQA requirements are found in NASA 8739.8, *Standard for Software Assurance*. Also relevant are NASA STD 8719.13B, *Software Safety*, and NASA-GB-8719.13, *NASA Software Safety Guidebook.*

D.9. STANDARDS FOR DOE QA RULE IMPLEMENTATION

ASME NQA-1-2102, Subpart 4.1.2 provides a crosswalk between ASME NQA-1 2008, 10 CFR Part 830 Subpart A, and DOE O 414.1C [97].

[97] DOE O 414.1C, Quality Assurance, dated 6-17-05

APPENDIX E. SAFETY SOFTWARE CRITERIA REVIEW AND APPROACH DOCUMENT (CRAD)

E.1 INTRODUCTION

This Appendix contains criteria and guidelines for assessing safety software used in Department of Energy (DOE) facilities. DOE Order (O) 414.1D, *Quality Assurance* requires that all software meet the applicable quality assurance (QA) requirements in Attachment 2, using a graded approach. The assessment criteria and guidance provided in this Appendix may also be applied to other software (OSW).

This document is organized as follows.

- The ***Assessment Guidelines*** section covers the purpose, scope, guiding principles, and assessment methodology.
- The ***Criteria and Approach*** section presents the objective, criteria, approach, and tailoring for the following work activities: (1) software project management and quality planning, (2) software risk management, (3) software configuration management, (4) procurement and supplier management, (5) software requirements identification and management, (6) software design and implementation, (7) software safety analysis and safety design methods, (8) software verification and validation, (9) problem reporting and corrective action, (10) training of personnel in the design, development, use, and evaluation of software.

E.2 PURPOSE AND SCOPE

The purpose and scope of this CRAD is to provide a set of consistent criteria and guidelines for the assessment of software, to ensure that software used in DOE's facilities is adequate. The SQA criteria for evaluating software are based on DOE O 414.1D, ASME NQA-1-2008/NQA-1a-2009, *Quality Assurance Program for Nuclear Facility Applications* and applicable Institute of Electrical and Electronics Engineers (IEEE) standards.

This CRAD is detailed enough to evaluate the overall quality assurance program (QAP) as it applies to software. It focuses rather on the software application or product and tailors the scope of the assessment to suit the specific software usage. The CRAD could be used for assessment of the following types of software:

- custom software developed by DOE, its contractors, or subcontractors;
- configurable software such as programmable logic controllers (PLCs); and
- acquired software such as commercial off the shelf (COTS) software.

E.3 GUIDING PRINCIPLES

The following principles should guide the conduct of the assessment. The assessment team leader, with assistance from the DOE site manager responsible for these assessments, should ensure that these guiding principles are incorporated in the tailoring process for assessing safety software and safety-affecting software applications.

- The team should review any previous assessments and reviews of the software. This review will enable the team to understand previous assessments, software qualification processes, associated requirements and performance criteria, assumptions concerning system operations, and the role of software in operations.
- The team should review any lessons learned from past events associated with software applications and include any additional attributes as appropriate in the assessment plan.
- The review of SQA processes for existing safety software should follow the guidance provided in Section 3.1.4, *Legacy Software Applications*.
- The physical boundaries of the software within the system or subsystem level or portions thereof under review should be documented in the assessment report.
- The assessment of specific software applications should begin with gaining an understanding of the overall system, then documenting the system functions, the performance criteria that the system should meet to successfully accomplish its functions, and the role of the software in ensuring that these functions and criteria are met. The potential consequences of failure of the software and the associated effects on system operability should be understood and documented.
- The facility staff should assist the team in understanding the associated software quality assurance (SQA) process. The staff should also provide documented evidence to the team that the appropriate SQA standards were applied to software development, procurement, or use.
- Procedures and records for software design, implementation, procurement, verification and validation, testing, and maintenance should be evaluated for adequacy.
- If the team identifies a condition that poses an imminent threat to personnel or facility safety, it should notify line management immediately. Team personnel should point out the imminent threat condition to their points of contact or appropriate facility manager, and notify the assessment team leader as soon as practical.
- Applicability of the assessment criteria and guidelines should be appropriate for the assessment scope. These assessment criteria and guidelines should be tailored to allow the most cost-effective use of resources and should take into account considerations such as recently completed assessments, evaluations, studies, inspections, and other relevant factors.

- The team should consider the type of software (custom-developed, configurable or acquired) when evaluating the adequacy of the SQA processes.
- The assessment should consider the effectiveness of SQA processes that are separate from system quality processes. In many instances, especially with acquired software, the separation of software from the system may increase costs but not increase the safe operation of the system.
- Information for existing software may be contained in other system-related documentation where software has an application.

E.4 CRITERIA AND APPROACH

The Criteria and Approach section is divided into the following work activities:

1. Software Project Management and Quality Planning
2. Software Risk Management
3. Software Configuration Management (SCM)
4. Procurement and Supplier Management
5. Software Requirements Identification and Management
6. Software Design and Implementation
7. Software Safety Analysis and Safety Design Methods
8. Software Verification and Validation (V&V)
9. Problem Reporting and Corrective Action
10. Training of Personnel in the Design, Development, Use, and Evaluation of Software

Each of these work activities includes the following:

- ***Objective:*** Describes the assessment objective for the work activity and the intended contribution to the adequacy of the software.
- ***Criteria:*** Suggests characteristics of the software that should be verified.
- ***Approach:*** Suggests information needed to guide the team in assessing the quality of the software. However, the team may choose to select another approach to meet the assessment-specific needs.

E.4.1 SOFTWARE PROJECT MANAGEMENT AND QUALITY PLANNING

Objective:

Software project management and quality planning should depict the organizational structure that supports software life-cycle stages and deliverables and influences and controls the quality of the software.

Criteria:

1. Software project management and quality planning have been implemented, depicting organizational structure, responsibilities, and authorities for those managing, performing, and assessing the software projects.
2. SQA activities, software practices, and documentation are periodically assessed.
3. Software quality activities have been effectively implemented.

Approach:

Identify how project management and QA planning are done for the project or facility at hand. Including:

- software project schedule;
- software project scope;
- software engineering activities, including software requirements and design;
- software V&V activities, including reviews and test;
- SCM activities;
- software risk management approach;
- software safety analysis and planning;
- supplier control;
- user and software staff training,
- standards, practices, conventions, and metrics;
- records and document collection, maintenance, and retention; and
- problem reporting and corrective action methods.

Many of the items listed above may be detailed in unconventional locations. Software V&V, for example, may be described in software test plans.

Determine whether the documents containing the software project management and QA plan are configuration-controlled and document-controlled, and that such controls will be maintained until the software is retired. Verify that the software project management and QA plan is reviewed and updated, as necessary, for completeness and consistency.

E.4.2 SOFTWARE RISK MANAGEMENT

Objective:

Software risk management seeks to prevent software from adversely affecting project risk.

Criteria:

1. Potential software risks identified
2. Likelihood and consequences of safety software failure determined.
3. Risks prioritized.
4. Risk avoidance, mitigation, and/or transfer strategies created.
5. Risks monitored.

Approach:

Identify how software risk management is accomplished at the site or facility. The risk management plan may be described in a standalone document or may be embedded in another document. Ensure that risk management planning covers:

- scope of the risk management activities;
- risk management policies and process (technical and managerial) under which risk management is to be performed;
- technical and managerial risks and likelihood and potential safety consequences;
- risk thresholds for the software application;
- risk avoidance, mitigation, or transfer options; and
- management techniques to address risks throughout project life-cycle, including tracking, decision-making, and feedback points.

E.4.3 SOFTWARE CONFIGURATION MANAGEMENT

Objective:

Software configuration is defined, maintained, and controlled until the software is retired.

Criteria:

1. Software configuration items are identified, baselined and controlled.
2. A baseline labeling system is established and implemented.
3. For custom-developed software, periodic configuration audits and reviews are conducted and documented.
4. Proposed software changes are documented, evaluated, and approved.
5. Only approved changes are implemented.

Approach:

Review appropriate documents, such as applicable procedures related to software change control, to determine if an SCM process exists and is effective. This determination is made based on the following actions:

- Verify the existence of documented processes to control, uniquely identify, describe, and document the configuration of each version or update of safety software and its related documentation. This documented evidence may be in the SCM plan or in another software or system level document.
- Verify that a configuration baseline is defined and that it is being adequately controlled. This baseline should include operating system components, any associated runtime libraries, acquired software executables, custom-developed source code files, users' documentation, the appropriate documents containing software requirements, software design, software V&V procedures, test plans and procedures, and any software development and quality planning documents.
- Verify that a baseline labeling system has been created that uniquely identifies each configuration item, identifies changes to configuration items by revision, and provides the ability to uniquely identify each configuration.
- Review procedures governing change management for installing new versions of the software components, including new releases of acquired software.
- Review software change packages and work packages to ensure that (1) possible impacts of software modifications are evaluated before changes are made, (2) various software system products are examined for consistency and revised as necessary after changes are made and updated, (3) software is tested according to established standards after changes have been made, (4) changes are evaluated and approved for release by the responsible organization, and (5) software verification activities are performed as necessary to ensure that the change does not adversely affect the performance of the software.
- Interview a sample of cognizant line managers for engineering and QA, and other personnel to verify their understanding of the change control process and their commitment to manage changes in a formal, disciplined, and auditable manner.
- Perform audits or reviews as necessary.

E.4.4 PROCUREMENT AND SUPPLIER MANAGEMENT

Objective:

Acquired software meets the appropriate level of QA, based on risk, safety, facility life-cycle, complexity, and project quality requirements.

Criteria:

1. Procurement documents identify the technical and quality requirements.
2. Acquired software meets the technical and quality requirements.
3. Suppliers' QA programs meet or exceed the QA requirements specified in the procurement documents.
4. Procurement documents specify supplier reporting of software defects to the purchaser and the purchaser's reporting of defects to the supplier.

Approach:

Suppliers of acquired software are evaluated to ensure that the safety software is developed under an appropriate QA program and satisfies the specific requirements. The assessment of software procurement process should include the following.

- Locate and examine software technical and QA requirements. These requirements may be embedded in the DOE contractor's or subcontractor's procurement document, software or system design description, or SQA plan. If not documented in the procurement contract, ensure that the supplier has received such technical and QA requirements.
- Verify that the supplier's QA program has been reviewed and meets or exceeds the procurement specification requirements. The purchaser may review the supplier's QA program through supplier assessment, supplier self-declaration, third-party certification, or other similar methods.
- Review evidence that the acquired software was evaluated for the appropriate level of quality. This evidence may be included in the test results, a test summary, supplier site visit reports or supplier QA program assessment reports.
- Review procurement or other documents between the supplier and purchaser for a documented process to report software defects from the supplier to the purchaser and the purchaser to the supplier.

E.4.5 SOFTWARE REQUIREMENTS IDENTIFICATION AND MANAGEMENT

Objective:

Software functions, requirements, and their bases are defined, documented and managed throughout the safety software life-cycle.

Criteria:

1. The software requirements are documented and consistent with software functions.
2. The functionality, performance, security, interface, and safety requirements for the safety software are complete, correct, consistent, clear, testable, and feasible.

3. The documented software requirements are controlled and maintained. Changes to the software requirements are reflected in any and all documentation.
4. Each requirement is uniquely identified and defined such that it can be objectively verified and validated.

Approach:

Review appropriate documents, such as documented safety analysis, safety analysis reports, technical safety requirements, procurement specifications and any system documentation to determine if the software requirements document is consistent with the system design. The software requirements may exist either as a standalone document, such as a software requirements specification, or may be embedded in other system or software documents.

- The assessment of the software identification process should include the following. Verify that the software requirements address functionality, performance, security, design inputs, design constraints, installation considerations, operating systems (if applicable), and external interfaces necessary to design the software exist and are documented.
- If access to the system by only authorized users is a requirement, verify that use of software is controlled so that only personnel on authorized user lists apply or maintain safety software.
- Verify that the software requirements are correct, unambiguous, complete, consistent, verifiable, modifiable and traceable as appropriate.
- Verify that acceptance criteria are established in the software requirements for each of the identified requirements. Such criteria should be used for V&V planning and performance as defined in each related life-cycle phase.
- Verify that the software requirement documents are controlled under the configuration change control and document control processes.
- Verify that software requirement documents are reviewed and updated as necessary.

E.4.6 SOFTWARE DESIGN AND IMPLEMENTATION

Objective:

The software design depicting the logical structure, information flow, logical processing steps, data structures and interfaces, is defined, documented and properly implemented.

Criteria:

1. The design, including interfaces and data structures, is correct, consistent, clearly presented, and feasible.
2. The design is completely and appropriately implemented in the software.
3. The design requirements are traceable throughout the software life-cycle.

Approach:

Review the appropriate documents, including design documents, review records, and source code listings. The design may be documented in a standalone document or embedded in other documents.

The software design description should contain the following information:

- A description of the major components of the software design as they relate to the software requirements, and any interactions with components.
- A technical description of the software with respect to control flow, control logic, mathematical model, data structure and integrity, and interface.
- A description of inputs and outputs including allowable or prescribed ranges for inputs and outputs.
- A description of error handling strategies and the use of interrupt protocols.
- The design described in a manner suitable for translating into computer codes.
- Evidence of reviews of the design and code for the appropriate grading exists. This may overlap with the software V&V work activity.
- Evidence of developer testing including any independent testing for the appropriate grading exists.

E.4.7 SOFTWARE SAFETY ANALYSIS AND SAFETY DESIGN METHODS

Objective

The design of the software components is developed in a manner that ensures the software modules will perform their intended functions in a consistent manner under design basis conditions.

Criteria:

1. Software systems are analyzed at the component level to ensure adequate safeguards are implemented to eliminate or mitigate the potential occurrence of a software defect that could cause a system failure.
2. Software is designed with simplicity and isolation of safety functions.
3. Where appropriate, fault tolerance and self-diagnostics are implemented in the software design.

Approach:

- Review hazard analysis documents to ensure that software component and interface failures are included. This analysis may be part of a software or system level failure modes and effects analysis, fault-tree analysis, event-tree analysis or other similar analyses.

- Review how the identified hazards are resolved. Various methods are used for hazards resolutions, such as eliminations, reduction of exposure, and controlling or minimizing the effects of a hazard.
- Confirm that the hazard analysis is periodically reassessed throughout the software life-cycle and that changes are incorporated as appropriate.
- For high consequence safety software, sample software modules for proof of design complexity evaluation and isolation of safety functions from non-safety functions.
- For high consequence safety software, where modules defects could impact the safe operation of the system, evaluate the software design for the implementation of fault tolerant and/or self-diagnostics techniques.

E.4.8 SOFTWARE VERIFICATION AND VALIDATION

Objective:

The V&V process (including acceptance testing) and related documentation for software are defined and maintained to ensure that (1) the software correctly performs all its intended functions; and that (2) the software does not perform any adverse unintended function.

Criteria:

1. Software deliverables have been verified, and validated for correct operation using reviews, inspections, assessments, observation, and testing techniques.
2. Relevant abnormal conditions have been evaluated for mitigating unintended functions through testing, observation, or inspection techniques.
3. Traceability of software requirements to software design and acceptance testing has been performed.
4. New versions of the software are verified and validated to ensure that the software meets the requirements and does not perform any unintended functions.
5. V&V activities are performed by competent staff other than those who developed the item being verified or validated. This may overlap with the training work activity.

Approach:

Review appropriate documents, such as SQA plans, review plans, walkthrough records, peer review records, desk check records, inspection reports, test plans, test cases, test reports, system qualification plans and reports, and supplier qualification reports to determine whether:

- Management processes exist for performing V&V and management and independent technical reviews;

- Reviews and inspections of the software requirement specifications, procurement documents, software design, code modules, test results, training materials, and user documentation have been performed by staff other than those who developed the item;
- Software design was performed prior to the software being used in operations;
- For design V&V:
  - Results of the software V&V are documented and controlled;
  - V&V methods include any one or a combination of design reviews, alternate calculations, and tests performed during program development; and
  - The extent of V&V methods chosen are a function of (1) the complexity of the software; (2) the degree of standardization; (3) the similarity with previously proved software; and (4) the importance to safety; and
- For test V&V:
  - Documentation for development, factory or acceptance testing, installation, and operations testing exists;
  - Documentation includes test guidelines, test procedures, test cases including test data, and expected results;
  - Results documentation demonstrates successful completion of all test cases or the resolution of unsuccessful test cases and proves direct traceability between the test results and specified software design;
  - Test V&V activities and their relationship with the software life-cycle are defined;
  - Software requirements and system requirements are satisfied by the execution of integration, system and acceptance testing;
  - Acceptable methods for evaluating the software test case results include (1) analysis without computer assistance, (2) other validated computer programs, (3) experiments and test, (4) standard problems with known solutions, and (5) confirmed published data and correlations;
  - Traceability exists from software requirements to design and testing, and if appropriate, to user documentation; and
  - Hardware and software configurations pertaining to the test V&V are specified.

E.4.9 PROBLEM REPORTING AND CORRECTIVE ACTION

Objective:

Formal procedures for software problem reporting and corrective actions for software errors and failures are established, maintained, and controlled.

Criteria:

1. Documented practices and procedures for reporting, tracking, and resolving problems or issues are defined and implemented.
2. An evaluation process exists for determining if the reported problem is a software defect, error, or something else.
3. Organizational responsibilities for reporting issues, approving changes, and implementing corrective actions are identified and found to be effective.
4. For software defects and errors, the defect or error is correlated with the appropriate software engineering elements, identified for potential impact, and all users are notified.
5. For acquired software, procurement documents identify the requirements to both the supplier and purchaser to report problems to each other.

Approach:

Review documents and interview facility staff for the problem reporting and notification process to determine whether

- A formal procedure exists for software problem reporting and corrective action development that addresses software errors, failures, and resolutions;
- Problems that impact the operation of the software are promptly reported to affected organizations;
- Corrections and changes are evaluated for impact and approved prior to being implemented;
- Corrections and changes are verified for correct operation and to ensure that no side effects were introduced;
- Preventive measures and corrective actions are provided to affected organizations in a timely manner; and
- The organizations responsible for problem reporting and resolution are clearly defined.

E.4.10 TRAINING OF PERSONNEL IN THE DESIGN, DEVELOPMENT, USE, AND EVALUATION OF SOFTWARE

Objective:

Personnel are trained/qualified and capable of performing assigned work. Continuing training to personnel is provided to maintain job proficiency.

Criteria:

1. A training or indoctrination program exists for each of the following:

- Software analysis;
- Software development (concept to retirement);
- Operations and use; and
- Assessment or evaluation of software.

2. The training/indoctrination provides for continuing education and training to improve performance and proficiency.
3. Training/indoctrination is commensurate with the scope, complexity, and importance of the tasks and the education, experience, and proficiency of the person.

Approach:

- Review training records or other documentation and conduct interviews to confirm a training or indoctrination program exists for each of the personnel assignments listed above.
- Verify the training program provides for continuing education.
- Verify the training program is adequate and appropriate for the scope, complexity, and importance of the task being performed.

APPENDIX F. ACRONYMS AND DEFINITIONS

F.1 ACRONYMS

ACE	abnormal conditions and events
ANS	American Nuclear Society
ANSI	American National Standards Institute
ASIC	application specific integrated circuits
ASME	American Society of Mechanical Engineers
ASQ	American Society for Quality
CAD	computer aided design
CCFD	Critical Characteristics for Design
CFR	Code of Federal Regulations
CGD	Commercial Grade Dedication
CMMI	Capability Maturity Model Integration
CNSSI	Committee on National Security Systems Instruction
COTS	commercial off-the-shelf
CRAD	criteria review and approach document
CS	critical software
DEAR	Department of Energy Acquisition Regulation
DID	Defense in Depth
DNFSB	Defense Nuclear Facilities Safety Board
DOE G	U.S. Department of Energy Guide
DOE O	U.S. Department of Energy Order
DOE P	U.S. Department of Energy Policy
DOE	U.S. Department of Energy
DSA	documented safety analysis
EDD	embedded digital device
EPRI	Electric Power Research Institute
FIPS	Federal Information Processing Standard
FMEA	failure modes and effects analysis
GB	guide book
GS	general software
I&C	instrumentation and control
IAEA	International Atomic Energy Agency

ICS	Industrial Control Systems
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISMS	Integrated Safety Management System
ISA	International Society of Automation
ISO	International Organization for Standardization
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NNSA	National Nuclear Security Administration
NPH	natural phenomena hazard
NRC	Nuclear Regulatory Commission
OSW	other software
PLC	programmable logic controller
PLD	programmable logic devices
QA	quality assurance
QAP	quality assurance program
RTM	requirements traceability matrix
SAC	specific administrative control
SAE	Society of Automotive Engineers
SAS	Safety-affecting software
SC	safety class
SCAPA	Consequence Assessment and Protective Actions
S/CI	Suspect/Counterfeit item
SCM	software configuration management
SDD	software design description
SDP	software development plan
SDS	software design description
SEI	Software Engineering Institute
SG	safety guide
SHADS	safety and hazard analysis and design software
SMACS	safety management and administrative controls software
SME	subject matter expert
SPMP	software project management plan
SQA	software quality assurance

SQAP	software quality assurance plan
SQASG	Software Quality Assurance Support Group
SRS	software requirement specification
SS	safety significant
SSC	structure, system, and component
SSCM	secure software configuration management
SSQA	safety software quality assurance
SSS	safety system software
SW-CMM	Software Capability Maturity Model
TR	technical report
TSR	technical safety requirement
V&V	Verification and Validation

F.2 DEFINITIONS

The following definitions are included with this Guide for convenience and clarification. DOE O 414.1D definitions shall take precedence over those included in this Appendix.

Acceptance Testing. The process of exercising or evaluating a system or system component by manual or automated means to ensure that it satisfies the specified requirements and to identify differences between expected and actual results in the operating environment. Source: ASME NQA-1-2008/NQA-1a-2009.

Administrative Controls. The provisions relating to organization and management, procedures, record keeping, assessment, and reporting necessary to ensure safe operation of a facility. Source: 10 CFR Part 830.

Assessment. A review, evaluation, inspection, test, check, surveillance, or audit, to determine and document whether items, processes, systems, or services meet specified requirements and perform effectively. Source: DOE O 414.1D.

Baseline. A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for use and further development, and that can be changed only by using an approved change control process. Source: ASME-NQA-1-2008/NQA-1a-2009.

Configuration Management. The process of identifying and defining the configuration items in a system (i.e., software and hardware), controlling the release and change of these items throughout the system’s life cycle, and recording and reporting the status of configuration items and change requests. Source: ASME NQA-1-2008/NQA-1a-2009.

Configuration Item. A collection of hardware or software elements treated as a unit for the purpose of configuration control. Source: IEEE Std 610.12-1990.

Consequence. An outcome of an event, hazard, threat, or situation. Source: IEEE Std 1540-2001.

Critical Characteristics. Important design, material, and performance characteristics of a commercial grade item or service that, once verified, will provide reasonable assurance that the item or service will perform its intended safety function. Source: ASME NQA-1a-2009, Subpart 2.14.

Design Authority. The engineer designated by the Acquisition Executive to be responsible for establishing the design requirements and ensuring that design output documentation appropriately and accurately reflect the design basis. The Design Authority is responsible for design control and ultimate technical adequacy of the design process. These responsibilities are applicable whether the process is conducted fully in-house, partially contracted to outside organizations, or fully contracted to outside organizations. The Design Authority may delegate design work [authorities] but not its responsibilities. Source: DOE O 413.3B.

Error. A condition deviating from an established baseline, including deviations from the current approved computer program and its baseline requirements. Source: ASME-NQA-1-2008/NQA-1a-2009.

Firmware. The combination of a hardware device, computer programs, and data that resides as read-only software on that device. Firmware is sometimes referred to as embedded software.

Functional Configuration Audit. An audit conducted to verify that the development of a configuration item has been completed satisfactorily, that the item has achieved the performance and functional characteristics specified in the functional or allocated configuration identification, and that its operational and support documents are complete and satisfactory. Source: IEEE Std 610.12 1990.

Graded Approach. The process of ensuring that the level of analyses, documentation, and actions used to comply with requirements are commensurate with:

- the relative importance to safety, safeguards, and security;
- the magnitude of any hazard involved;
- the life-cycle stage of a facility or item;
- the programmatic mission of a facility;
- the particular characteristics of a facility or item;

- the relative importance to radiological and nonradiological hazards; and
- any other relevant factors.
  Source: 10 CFR Part 830.

Hazard Controls. Measures to eliminate, limit, or mitigate hazards to workers, the public, or the environment, including:

- physical, design, structural, and engineering features;
- safety structures, systems and components;
- safety management programs;
- Technical Safety Requirements; and
- other controls necessary to provide adequate protection from hazards.
  Source: 10 CFR Part 830.

Item. An all-inclusive term used in place of appurtenance, assembly, component, equipment, material, module, part, structure, product, subassembly, subsystem, system, unit, or support systems. Source: 10 CFR Part 830.

Nuclear Facility. A reactor or a nonreactor nuclear facility where an activity is conducted for or on behalf of DOE and includes any related area, structure, facility, or activity to the extent necessary to ensure proper implementation of the requirements established in CFR, part 10, section Part 830. Source: 10 CFR Part 830.

Physical Configuration Audit. An audit conducted to verify that a configuration item, as built, conforms to the technical documentation that defines it. Source: IEEE Std 610.12-1990.

Process. A series of actions that achieves an end result. Source: 10 CFR Part 830.

Quality. The condition achieved when an item, service, or process meets or exceeds the user's requirements and expectations. Source: 10 CFR Part 830.

Quality Assurance. All those actions that provide confidence that quality is achieved. Source: 10 CFR Part 830.

Quality Assurance Program. The overall program or management system established to assign responsibilities and authorities, define policies and requirements, and provide for the performance and assessment of work. Source: 10 CFR Part 830.
Risk. The likelihood of an event, hazard, threat, or situation occurring and its undesirable consequences; a potential problem. Source: IEEE Std 1540-2001.

Safety. An all-inclusive term used synonymously with environment, safety, and health to encompass protection of the public, the workers, and the environment. Source: DOE O 414.1C.

Safety-class structures, systems, and components (SC SSCs). Structures, systems, or components, including portions of process systems, whose preventive and mitigative function is necessary to limit radioactive hazardous material exposure to the public, as determined from the safety analyses. Source: 10 CFR Part 830.

Safety-significant structures, systems, and components (SS SSCs). Structures, systems, and components which are not designated as safety-class SSCs, but whose preventive or mitigative function is a major contributor to defense in depth and/or worker safety as determined from safety analyses [10 CFR Part 830].

Safety Management Program. A program designed to ensure a facility is operated in a manner that adequately protects workers, the public, and the environment by covering a topic such as: quality assurance; maintenance of safety systems; personnel training; conduct of operations; inadvertent criticality protection; emergency preparedness; fire protection; waste management; or radiological protection of workers, the public, and the environment. Source: 10 CFR Part 830.

Safety Software. Includes the following: Source: DOE O 414.1D

> **Safety System Software**. Software for a nuclear facility that performs a safety function as part of an SSC and is cited in either (a) a DOE-approved documented safety analysis; or, (b) an approved hazard analysis per DOE P 450.4, *Safety Management System Policy*, dated 10-15-96 (or latest version) and 48 CFR 970-5223.1.
>
> **Safety and Hazard Analysis Software and Design Software**. Software that is used to classify, design, or analyze nuclear facilities. This software is not part of an SSC but helps to ensure the proper accident or hazards analysis of nuclear facilities or an SSC that performs a safety function.
>
> **Safety Management and Administrative Controls Software**. Software that performs a hazard control function in support of nuclear facility or radiological safety management programs or technical safety requirements or other software that performs a control function necessary to provide adequate protection from nuclear facility or radiological hazards. This software supports eliminating, limiting, or mitigating nuclear hazards to workers, the public, or the environment as addressed in 10 C.F.R. Parts Part 830 and 835, the DEAR Integrated Safety Management System clause, and 48 CFR 970-5223.1.

Safety Structures, Systems, and Components. Both safety class structures, systems, and components and safety significant structures, systems, and components. Source: 10 CFR Part 830.

Service. Work, such as design, construction, fabrication, decontamination, environmental remediation, waste management, laboratory sample analysis, safety software development/ validation/testing, inspection, nondestructive examination/testing, environmental qualification, equipment qualification, training, assessment, repair, and installation or the like. Source: 10 CFR Part 830.

Software. Computer programs and associated documentation and data pertaining to the operation of a computer system. Source: ASME-NQA-1-2008/NQA-1a-2009.

Software Life-cycle. The activities that comprise the evolution of software from conception to retirement. The software life cycle typically includes the software development cycle and the activities associated with operation, maintenance, and retirement. Source: ASME-NQA-1-2008/NQA-1a-2009.

Technical Safety Requirements. The limits, controls, and related actions that establish the specific parameters and requisite actions for the safe operation of a nuclear facility and include, as appropriate for the work and the hazards identified in the documents safety analysis for the facility: safety limits, operating limits, surveillance requirements, administrative and management controls, use and application provisions, and design features, as well as a bases appendix. Source: 10 CFR Part 830.

Validation. The process of: (a) evaluating a system or component during, or at the end of the development process to determine whether it satisfies specified requirements; or, (b) providing evidence that the software, and its associated products, satisfies system requirements allocated to software at the end of each life-cycle activity, solves the right problem (e.g., correctly models physical laws, implements business rules, uses the proper system assumptions), and satisfies the intended use and user needs. Source: DOE O 414.1D.

Verification. The process of: (a) evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase; or, (b) providing objective evidence that the software and its associated products conforms to requirements (e.g., for correctness, completeness, consistency, accuracy) for all life-cycle activities during each life-cycle process (acquisition, supply, development, operation, and maintenance); satisfies standards, practices, and conventions during life-cycle processes; and, successfully completes each life-cycle activity and satisfies all the criteria for initiating succeeding life-cycle activities (e.g., building the software correctly). Source: DOE O 414.1D.

Work. A defined task or activity; such as: research and development; operations; environmental remediation; maintenance and repair; administration; software (including safety software) development, validation, testing, and use; inspection; safeguards and security; or data collection and analysis. Source: DOE O 414.1D.

APPENDIX G. REFERENCES

The following referenced documents were used in developing the information contained in this Guide. Some of these documents, such as Department of Energy (DOE) Orders and the 10 CFR Part 830, Part A, *Quality Assurance*, may be obtained through the online DOE Directives, Regulations, and Standards website: http://www.directives.doe.gov. Other documents, such as the American Society of Mechanical Engineers (ASME), American Society for Quality (ASQ), and International Electrotechnical Commission (IEC) standards and guidance documents may be purchased or obtained from the sponsoring organizations.

1. 10 CFR Part 830, *Nuclear Safety Management*.

2. DEAR 48 CFR §970.5223-1, *Integration of environment, safety, and health into work planning and execution.*

3. American National Standards Institute (ANSI)/American Nuclear Society (ANS) 10.4-1987 (R1998), *Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry*, American Nuclear Society, 1998.

4. ANSI/ANS 10.7-2013, *Non-Real-Time, High-Integrity Software for the Nuclear Industry—Developer Requirements*, ANSI/ANS, March 2013.

5. ANSI/International Society of Automation (ISA) 84.00-01, *Functional Safety: Safety Instrumented Systems for the Process Industry Sector* (2004).

6. American Society of Mechanical Engineers (ASME) NQA-1-2008/NQA-1a-2009, *Quality Assurance Program for Nuclear Facilities*.

7. ASME NQA-1-2008/2009A Part II, Subpart 2.7, *Quality Assurance Requirements for Computer Software for Nuclear Facility Applications*.

8. ASME NQA-1a-2009, Part II, Subpart 2.14, *Quality Assurance Requirements for Commercial Grade Items and Services*.

9. ASME NQA-1-2012 Part 3.2-2.14, *Implementing Guidance for Part II, Requirement 2.14: Quality Assurance Requirements for Commercial Grade Items and Services, Commercial Grade Computer Programs, and Software Services*.

10. ATEC, *Test and Evaluation, Modeling and Simulation Verification, Validation, and Accreditation Methodology*, Pamphlet 73-21, Department of the Army, Army Test and Evaluation Command, Alexandria, VA, April 2007.

11. Capability Maturity Model Integration (CMMI) Product Team, *Capability Maturity Model Integration*, Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1),

Continuous Representation, CMU/SEI-2002-TR-011, ESC-TR-2002-011, Carnegie Mellon University (CMU) Software Engineering Institute (SEI), 2002.

12. Chew, Jennifer, and Cindy Sullivan, *Verification, Validation, and Accreditation in the Life Cycle of Models and Simulations*, Proceedings of the 2000 Winter Simulation Conference, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, Orlando, FL, December, 2000, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7220.

13. Christensen, Mark J., and Richard H. Thayer, *The Project Manager's Guide to Software Engineering's Best Practices,* Institute of Electrical and Electronics Engineers, Computer Society Press, 2001.

14. Committee on National Security Systems Instruction (CNSSI) No. 1253 "Security Categorization and Control Selection for National Security Systems," March 27, 2014.

15. DA, *Verification, Validation, and Accreditation of Army Models and Simulations*, Pamphlet 5-11, Department of the Army, Washington, DC, September 1999.

16. Defense Nuclear Facilities Safety Board (DNFSB) Recommendation 2002-1, *Quality Assurance for Safety-Related Software*, Defense Nuclear Facilities Safety Board, September 2002.

17. DOE O 200.1A, *Information Technology Management*, dated 12-23-2008

18. DOE O 205.1B, *DOE Cyber Security Program*.

19. DOE O 414.1D, *Quality Assurance*, dated 4-25-2011. (Change Notice No. 1, dated May 2013).

20. DOE P 450.4A, *Integrated Safety Management Policy*.

21. DOE, *EM Quality Assurance Program*, EM-QA-001, Rev. 1, Office of Environmental Management, June 2012.

22. DOE Office of Environmental Management, *Guidance for Commercial Grade Dedication*, April 2011.

23. DOE Office of Health, Safety and Security Safety Advisory for Software Quality Assurance, October 2010 Advisory No. 2010-08: *Firmware Defect in Programmable Logic Controller.*DOE-STD-1027-92, *Hazard Categorization and Accident Analysis Techniques For Compliance With DOE Order 5480.23, Nuclear Safety Analysis Reports*, December 1992 (Change Notice No. 1, dated September 1997).

24. DOE-STD-1195, *Design of Safety Significant Safety Instrumented Systems Used at a DOE Nonreactor Nuclear Facilities.*

25. DOE-STD-3009-2014, *Preparation of Nonreactor Nuclear Facility Documented Safety Analysis,* November 2014.

26. DOE Subcommittee on Consensus Assessment and Protective Actions Special Interest Group (SCAPA), *SQA Guidance,* July 30, 2010.

27. EPA, *Guidance on the Development, Evaluation, and Application of Environmental Models,* EPA/100/K-09/003, Environmental Protection Agency, March 2009.

28. EPRI TR 1025243 Plant Engineering*: Guideline for the Acceptance of Commercial-Grade Design and Analysis Computer Programs Used in Nuclear Safety-Related Applications.*

29. ERPI TR-106439 Plant Engineering*: Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications.*

30. International Atomic Energy Agency (IAEA) Safety Guide (SG) Series No. NS-G-1.1, *Software for Computer Based Systems Important to Safety in Nuclear Power Plants,* IAEA, 2000.

31. IAEA Safety Series No. 50-C/SG-Q, *Quality Assurance for Safety in Nuclear Power Plants and other Nuclear Installations,* Code and Safety Guides Q1–Q14, IAEA, 1996.

32. IAEA Technical Reports Series No. 397, *Quality Assurance for Software Important to Safety,* IAEA, 2000.

33. Identifying software failures using FMEA and fault tree analysis, ASQ Quality Progress, September 2012.

34. IEC 61508 Parts 1–7, *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems,* IEC, 1998.

35. IEC 61511 Parts 1–3, *Functional Safety—Safety Instrumented Systems for the Process Industry Sector*, IEC, 2003.

36. IEEE/ISO/IEC 16326-2009, Systems and Software Engineering--Life Cycle Processes--Project Management.

37. IEEE Std 610.12-1990, IEEE *Standard Glossary of Software Engineering Terminology*, IEEE, 1990.

38. IEEE Std 730-2002, *Standard for Software Quality Assurance Plans*, IEEE, 2002.

39. IEEE Std 828-1998, *IEEE Standard for Software Configuration Management Plans,* IEEE, 1998.

40. IEEE Std 1012-1998, *IEEE Standard for Software Verification and Validation,* IEEE, 1998.

41. IEEE Std 1012a-1998, *IEEE Standard for Software Verification and Validation—Supplement to 1012,* IEEE, 1998.

42. IEEE Std 1016-1987, *IEEE Recommended Practice for Software Design Descriptions,* 1987.

43. IEEE Std 1042-1987, *IEEE Guide to Software Configuration Management,* Section 3.3.4, "Audits and Reviews," IEEE, 1987.

44. IEEE Std 1228-1994, *IEEE Standard for Software Safety Plans,* IEEE, 1994.

45. IEEE Std 7-4.3.2-2003, *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations,* IEEE, 2003.

46. IEEE Std 730-2002, IEEE Std 730-2014, *IEEE Standard for Software Quality Assurance Processes*, 2014.

47. IEEE Std 1540-2001, IEEE Standard for Software Life Cycle Processes - Risk Management, IEEE, 2001.

48. IEEE-Std-24765, *Systems and Software Engineering Vocabulary,*

49. IP 2002-1, Implementation Plan for Defense Nuclear Facilities Safety Board Recommendation 2002-1, *Quality Assurance for Safety-Related Software at Department of Energy Defense Nuclear Facilities,* dated 3-13-03.

50. ISO 9001-2008, *Quality Management Systems—Requirements*, ISO, 2008.

51. ISO/IEEE Standard 16085, *IEEE Standard for Software Engineering: Software Life Cycle Processes, Risk Management,* IEEE, 2004.

52. Leveson, Nancy, *Safeware: System Safety and Computers,* Addison Wesley, 1995.

53. M&SCO (2006), *VV&A Recommended Practices Guide*, Modeling and Simulation Coordination Office (M&S CO), *VV&A Recommended Practices Guide*, September 2006. (http://msco.mil/VVA_RPG.html)

54. National Aeronautics and Space Administration (NASA) NASA-STD- NASA-STD-8739.8, *Software Assurance Standard,* NASA, 2004.

55. NASA-GB-8719.13 *NASA Software Safety Guidebook*, NASA, 2004

56. NASA-STD-13B *Software Safety Standard,* NASA, 2004.National Aeronautics and Space Administration (NASA)

57. National Research Council, *Models in Environmental Regulatory Decision Making*, National Research Council, Washington D.C., 2007.

58. NIST 800-37, "Guide for Applying the Risk Management Framework to Federal Information Systems."

59. NIST 800-53 Revision 4 "Security and Privacy Controls for Federal Information Systems and Organizations."

60. NIST, *Reference Information for the Software Verification and Validation Process*, NIST Special Publication 500-234, National Institute of Standards and Technology, Gaithersburg, MD, March 1996.

61. NRC Information Notice No. 94-20: *Common-cause Failures due to Inadequate Design Control and Dedication.*

62. Overcamp, William L. and Timothy G. Trucano, *Verification and Validation in Computational Fluid Dynamics*, SAND2002-0529, Sandia National Laboratories, March 2002.

63. Roach, Patrick J., *Verification and Validation in Computational Science and Engineering,* Hermosa Publishers, Albuquerque, NM, 1998.

64. Sargent, R. G., *Verification and Validation of Simulation Models*, In Proc. 2011 Winter Simulation Conf., ed. S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers Inc., 2011.

65. Society of Automotive Engineers (SAE), SAE JA1003, *Software Reliability Program Implementation Guide*, SAE, 2004.

66. Sparkman, Debra, *Techniques, Processes, and Measures for Software Safety and Reliability,* Lawrence Livermore National Laboratory, UCRL-ID 108725, 1992.

67. SQAS21.01.00-1999 (Reference Document), *Software Risk Management: A Practical Guide,* Department of Energy Quality Managers Software Quality Assurance Subcommittee, February 2000.

68. SQASG TP-07-01 Rev 2, *DOE Safety Software Examples*.

69. SQASG-TP-10-01-REV. 1, *A Systematic Approach to Implementing the Quality Requirements of DOE O 414.1D for Software*.

70. SQASG-TP-14-01 REV. 0, *DOE Nuclear Safety Software Graded Approach Assessment Tool*.

71. Thacker, B. H., S. W. Doebling, F. M. Hemez, M. C. Anderson, J. E. Pepin, and E. A. Rodriguez, *Concepts of Model Verification and Validation*, LA-14167-MS, Los Alamos National Laboratory, October 2004.

72. U.S. Nuclear Regulatory Commission 10 CFR §73.54, "Protection of digital computer and communication systems and networks."

73. U.S. Nuclear Regulatory Commission, Regulatory Guide 5.71, "Cyber Security Programs for Nuclear Facilities," January 2010.