

Simplified Application of a Nuclear Graded Approach to Software Quality Assurance for Beginners

Laura Suddath © July 2026



Contents

Introduction.....	3
Purpose, Scope, and Applicability	4
Clarification: What a Graded Approach Does—and Does Not—Allow	4
The NQA-1 Software Landscape.....	5
Classification: Grading the Software	5
Decision Flow for Software Control.....	6
Control Scaling: How to Effectively Control Graded Software	6
Key Pitfalls and Minimum Controls	8
Framework for Applying a Graded Software Control Approach.....	8
Commercial Grade Dedication for Software	11
When CGD May Not Be Required	13
How Software CGD Works.....	13
Document and Approve the Dedication Package.....	15
Typical Quality Records and Evidence	15
Critical Software CGD Constraints	16
Combining CGD Methods	16
Conclusion	19
Acronyms and Key Definitions	21
References.....	24

Introduction

Software is used throughout nuclear work. It may perform design calculations, control digital equipment, record data, manage documents, or support engineering decisions. These tools do not all create the same level of risk. A program that directly supports a safety function needs stronger controls than a simple administrative database.

A **graded approach** matches the amount of software quality assurance work to the risk. Higher-risk software receives more documentation, testing, independent review, configuration control, and approval. Lower-risk software still remains controlled, but the controls may be simpler. This article explains the process in plain language: identify the software, decide how important it is, select the needed controls, verify that it works, approve its use, and control later changes.

Beginner's Roadmap

1. **List the software.** Record the name, version, owner, purpose, and where it is used.
2. **Define the job.** State exactly what the software is expected to do and what decisions depend on its output.
3. **Classify the risk.** Ask what could happen if the software fails or gives a wrong answer.
4. **Choose the controls.** Apply stronger controls to higher-risk software and simpler controls to lower-risk software.
5. **Verify performance.** Test the software, independently check results, or use an approved acceptance method.
6. **Approve and control use.** Record the accepted version, configuration, environment, users, and limits of use.
7. **Review every change.** Evaluate patches, upgrades, platform changes, and new uses before continued reliance.

Simple rule: The greater the consequence of a wrong result, the stronger the controls and evidence should be.

Purpose, Scope, and Applicability

This article is intended for nuclear quality assurance programs that control software used in safety-related, safety-affecting, or quality-affecting applications. It is most applicable to nuclear suppliers, design organizations, Department of Energy facilities, advanced reactor developers, and commercial nuclear organizations that implement ASME NQA-1, 10 CFR 50 Appendix B, 10 CFR Part 21, or customer-imposed nuclear QA requirements.

ASME NQA-1:2015, contract requirements, licensing basis, and implementing procedures should always govern final application. This article provides a practical framework for applying a graded approach; it does not replace project-specific procedures, procurement requirements, regulatory commitments, or customer-approved quality program requirements.

Clarification: What a Graded Approach Does—and Does Not—Allow

A graded approach does **not** remove applicable quality requirements. It is a documented way to decide how much rigor is needed. Consider the software's function, safety importance, complexity, possible failure consequences, and how easily an error would be detected. Record the classification decision, the selected controls, and the evidence showing that the software is fit for its approved use.

For commercial software, also control procurement, versions, changes, reported problems, and—when required—Commercial Grade Dedication (CGD). Before using a commercial program for a safety-related decision, clearly define the intended use, accepted version, operating environment, important performance features, acceptance method, and maintenance controls.

The NQA-1 Software Landscape

Under the standard, software controls are governed by two distinct components:

1. **Part I, Requirement 3, Section 800 (Software Design Control):** Outlines overall design, testing, and lifecycle mandates.
2. **Part II, Subpart 2.7 (Quality Assurance Requirements for Computer Software):** Details explicit programmatic requirements for development, verification and validation (V&V), configuration management, and commercial-grade dedication.

Classification: Grading the Software

Start by asking one question: **What could happen if this software fails or gives the wrong answer?** Use the answer to place the software into the organization's approved quality level or functional class. The labels may differ by organization, but the basic pattern is high, medium, or lower risk.

Table 1 -- Software Classification and NQA-1:2015 Control Rigor

Software Class	Risk / Impact Profile	Examples	NQA-1 Control Rigor
Safety-Related (Class 1 / QAL I)	Failure directly compromises reactor core integrity or triggers a public radiation hazard.	Reactor Protection Systems (RPS), Emergency Core Cooling actuation.	Full Subpart 2.7 compliance. Complete independent V&V required.
Safety-Affecting / Defense-in-Depth (Class 2 / QAL II)	Failure degrades a secondary barrier or provides critical engineering data for design safety.	Transient safety analysis codes, stress calculation tools, radiation monitoring.	Tailored controls. Rigorous testing and document control; independent review.
Non-Safety / Quality-Affecting (Class 3 / QAL III)	Failure impacts production, commercial viability, or general plant efficiency, but no safety hazard.	Document control databases, inventory tracking, general CAD tools.	Standard QA. Basic functional testing and user verification; standard commercial practices.

Decision Flow for Software Control

Use Table 2 after classifying the software and before selecting detailed controls. Read each question from top to bottom. A “yes” answer points to the action in the middle column, and a “no” answer points to the action in the last column. Document each decision and the supporting evidence.

Table 2 -- Decision Flow for Software Control

Decision Point	If Yes	If No
Does the software perform or support a safety-related function?	Apply the highest applicable software QA controls and determine whether full Subpart 2.7 controls are required.	Evaluate whether the software is safety-affecting or quality-affecting and apply scaled controls.
Was the software developed under an approved nuclear QA program?	Control the software under the approved supplier or internal QA program and applicable procurement requirements.	Evaluate whether commercial-grade dedication or qualification of otherwise acquired software is required.
Is the output fully verified by an independent method before use?	Document the verification method and retain evidence supporting use within the approved scope.	Dedicate or qualify the software before relying on the output for safety-related or safety-affecting decisions.
Will the version, configuration, platform, or intended use change?	Perform an impact review and targeted regression testing before continued use.	Maintain the approved baseline and continue periodic review under configuration control.

Control Scaling: How to Effectively Control Graded Software

After classification, match the controls to the risk. Use the questions below as a beginner’s guide.

- **How much planning is needed?** High-risk software needs clearly separated and documented life-cycle stages. Lower-risk software may combine information in one controlled record.

- **Who checks the work?** The greater the risk, the more independent the verification should be.
- **How are versions controlled?** Identify the approved baseline and prevent unreviewed changes.
- **What evidence is retained?** Keep enough records to show what was tested, who approved it, and what limits apply.

Lifecycle Phase Management

- **High Rigor (Safety-Related):** Enforce distinct, fully documented life cycle phases. Each applicable phase (Requirements → Design → Implementation → Testing → Operation) should have defined entry and exit criteria appropriate to the software's safety significance, with independent peer approvals where required.
- **Low Rigor (Non-Safety):** Combine requirements and design steps into a single "Software Data Sheet" or functional description file to minimize administrative overhead.

Verification and Validation (V&V)

- **High Rigor:** Entrust V&V to individuals entirely independent of the original development team. Use advanced mathematical modeling, boundary testing, and structural code coverage utilities.
- **Low Rigor:** Rely on simpler functional tests or rely on historical vendor data alongside a simple "user-acceptance test" to confirm correct installation.

Configuration Management (CM)

- **High Rigor:** Maintain locked software baselines. Any modification to code, build environments, or requirements must pass through a formal Change Review Board (CRB) with exhaustive regression testing before deployment.
- **Low Rigor:** Rely on standard version control numbering systems. Log modifications locally without requiring executive board intervention.

Commercial-Grade Dedication (CGD)

If commercial off-the-shelf (COTS) software will be used in a safety-related application, evaluate the applicable requirements in **Part II, Subpart 2.14** and **Subpart 2.7, Section 302**.

The dedicated CGD section later in this article explains when dedication is needed and how to perform it.

- Identify the application's **Critical Characteristics** (e.g., calculation accuracy, processing speed).
- Validate those attributes through active dedication pathways, such as third-party commercial surveys, source verification, or extensive pre-operational site tests.

Key Pitfalls and Minimum Controls

Avoid three common mistakes. First, do not treat grading as permission to ignore applicable requirements. Second, do not approve software without clearly defining its allowed use. Third, do not allow patches, upgrades, or configuration changes without review. At a minimum, controlled software should have an identified version, an approved scope of use, appropriate access control, backup or recovery provisions, change control, and evidence that it is fit for its intended purpose.

Framework for Applying a Graded Software Control Approach

Functional Mapping Matrix

The software's job is the best starting point. A program that controls equipment, performs a safety calculation, or only records data will need different controls.

Use Table 3 to find the closest software function. Then use its typical risk level and engineering focus as a starting point. Confirm the final classification using your approved procedure, contract, licensing basis, and quality program.

Table 3 -- Functional Mapping Matrix for Software Applications

Function	Primary Software Type	Typical NQA-1 Classification	Primary Engineering Focus
Instrumentation & Control (I&C)	Internal Custom (Firmware, PLC)	High Rigor (Safety-Related / Class 1)	Deterministic execution, real-time safety, fault tolerance.
Design Calculations	COTS Tools (ANSYS, MATLAB, Excel)	Medium/High (Safety-Affecting / Class 2)	Algorithmic accuracy, boundary verification, numerical precision.

Data Logging	Hybrid (COTS SCADA + Custom DB)	Low/Medium (Quality-Affecting / Class 3)	Data integrity, secure storage, retrieval reliability.
---------------------	---------------------------------	---	--

Practical Execution Steps

1. **Create a Software Inventory Log (SIL):** Record every software application, version, configuration, owner, macro, purpose, and location of use.
2. **Classify the Software:** Document the function, safety significance, failure consequence, and approved quality level or functional class.
3. **Define the Scope of Use:** State the permitted users, functions, inputs, outputs, operating environment, and technical boundaries.
4. **Select and Perform the Controls:** Apply the required planning, verification and validation, configuration management, testing, acceptance, and approval activities.
5. **Maintain the Approved Baseline:** Review defects, patches, upgrades, platform changes, and expanded uses before continued reliance.

Implementation Control Framework

Control Area 1: Instrumentation and Control—Internal Custom Software

Because custom I&C code actively controls hardware and protects margins, it demands complete **Subpart 2.7** lifecycle compliance.

- **Requirements & Design:** Code must stem from explicit Functional Requirements Specifications (FRS). The organization should document a Software Architecture Design that shows how the system handles hardware exceptions, failures, or data corruption without violating safety boundaries.
- **Testing Rigor:** Manual testing alone is typically insufficient for high-rigor custom I&C applications. The organization should use hardware-in-the-loop (HIL) simulators, boundary testing, fault-injection scenarios, or other justified methods to stress-test the custom code under credible environmental and input failure conditions.
- **Independent V&V:** The engineer writing the ladder logic or C/C++ firmware **should not** be the person verifying it. V&V should be performed by an independent engineer or external QA group using independent test vectors.

Control Area 2: Design Calculations—COTS Tools

Because the organization typically cannot modify proprietary solver source code, such as MATLAB or ANSYS, controls shift from design lifecycle monitoring to dedication and validation under Subpart 2.7, Section 302, and Subpart 2.14.

- **Critical Characteristics:** Define exactly what matters to your calculations. For example, if using FEA software, the critical characteristics are the specific element types (e.g., solid, shell) and material matrix solvers you plan to deploy.
- **Black-Box Validation of Commercial Software:** Treat the tool as a black box by running standardized problem sets, such as benchmark problems with known, hand-calculated analytical solutions, within the software. If the software's output matches the analytical solution within an acceptable tolerance, it is dedicated for use *only* within those parameters.
- **User Controls:** Standardize macros and scripts. If an engineer designs an advanced Excel calculation sheet, that specific sheet must be locked, version-controlled, and verified as a custom software tool before production use.

Control Area 3: Data Logging—Hybrid or COTS Software

Data loggers generally record parameters rather than controlling them. The graded approach can reduce administrative overhead where risk and safety significance support the reduction.

- **Installation & Functional Testing:** Instead of exhaustive lifecycle audits, use an Installation Qualification (IQ) and Operational Qualification (OQ), or an equivalent documented verification approach. Confirm that the software installs correctly on the network and accurately logs test data points without dropping packets.
- **Data Integrity & Security:** Focus controls on configuration management. Restrict permissions so data-logging configurations cannot be modified without generating an alert or requiring an administrator login. Ensure automated, verified backups are active.

Lifecycle Activity Comparison Matrix

Lifecycle Element	Internal Custom I&C (High)	COTS Design Tools (Medium)	COTS/Custom Data Logging (Low)
Requirements	Formal software requirements	Documented user capabilities	Basic user scope
V&V Activity	Code reviews plus documented test coverage appropriate to safety significance	Benchmark Problem Set Testing	Functional Input/Output Checks
V&V Independence	Total structural independence	Technical review by an independent engineer	Self-verification or supervisor sign-off
Configuration Control	Software Change Review Board (CRB)	Version locking via IT/Admin controls	Local access control lists (ACL)
Retirement Plan	Formal data migration & archiving	Simple media decommissioning	Standard file/database archival

Commercial Grade Dedication for Software

Commercial Grade Dedication (CGD) for software is a formal acceptance process used to provide reasonable assurance that commercial off-the-shelf (COTS) software will perform its intended safety function reliably within its approved scope of use.

Under **ASME NQA-1:2015**, including the commercial grade dedication provisions in Part II, Subpart 2.14 and the software acquisition provisions in Subpart 2.7, Section 302, commercial off-the-shelf software should not be accepted for nuclear safety use based only on purchase, installation, or vendor reputation. When the software was not produced under an approved nuclear quality assurance program, the purchaser must establish documented acceptance evidence before relying on it for a safety-related or safety-affecting function. In practice, this dedication activity provides the basis for treating the commercial software as acceptable for its defined nuclear application.

Worked Example: COTS Structural Analysis Tool

Consider a commercial off-the-shelf (COTS) finite element analysis tool used to support seismic qualification of safety-related piping. Its classification depends on how directly its output supports the safety function. The organization should define a bounded scope of use, such as linear elastic structural analysis for specified materials, element types, loading conditions, and acceptance limits.

The dedication package should identify critical characteristics such as solver accuracy, input-boundary handling, version identification, operating environment, and output traceability. Method 1 benchmark testing can confirm calculation accuracy against known solutions, while Method 2 survey evidence can support dependability characteristics such as release control, defect tracking, and regression testing. Any version change, patch, operating-system change, or expanded analysis scope should trigger an impact review and targeted revalidation.

Why CGD for Software Is Used

Many commercial software products are developed for broad industrial, engineering, or business markets rather than for nuclear procurement under 10 CFR 50 Appendix B or an NQA-1 quality assurance program. As a result, the purchaser may receive useful technical capability without the nuclear-quality records, supplier controls, or auditable development evidence normally expected for safety-related applications.

For that reason, the purchaser or dedicating entity must create the technical basis for acceptance. The dedication package should show that the selected software, version, configuration, operating environment, and approved scope of use have been evaluated and that the critical characteristics needed for the intended safety function have been verified by appropriate acceptance methods.

When CGD for Software Is Used

Use this simple three-part test. CGD may be needed when **all three** answers are “yes”:

1. **Safety Significance:** The software will be used in a safety-related application, such as controlling a safety system, generating data used to design a safety system, or verifying safety margins.
2. **Commercial Origin:** The software was not developed under an approved nuclear QA program.
3. **Autonomous/Critical Output:** The software's output is relied upon directly, and errors would not be easily or immediately caught by a separate, routine human check.

Examples of Applications That May Require CGD

- **Design & Analysis Tools:** Using a COTS Finite Element Analysis (FEA) tool (like ANSYS) or a computational fluid dynamics tool to design a reactor pressure vessel or safety-tier piping.
- **Embedded Software:** Buying a commercial digital valve controller, PLC, or smart transmitter that contains pre-installed vendor firmware to regulate emergency cooling water flow.
- **Engineering Calculators:** Using MATLAB or advanced Excel configurations to compute nuclear fuel enrichment parameters or radiological shielding thickness.

When CGD May Not Be Required

- The software is purely administrative (e.g., standard scheduling, accounting, or general HR databases).
- The software's output is fully verified by an alternate method (e.g., an engineer uses software for a preliminary calculation but completely re-verifies the final results using manual hand-calculations).

How Software CGD Works

Software CGD can be understood as six steps: define the safety function, identify the critical characteristics, select acceptance methods, collect objective evidence, approve the dedication package, and control the accepted configuration. Critical characteristics are the

features that must be correct for the software to perform its approved safety function—for example, calculation accuracy, version identity, error handling, traceable output, and reliable operation in the approved environment.

Define the Intended Safety Function

The intended safety function should clearly state exactly what the software will do.

- *Example:* "The software must accurately compute structural stress limits under seismic loads between 0.1Hz and 100Hz."

Identify Critical Characteristics

The organization should select the specific capabilities that must perform correctly for the safety function to succeed.

- *Functional Attributes:* Correct mathematical matrix solving, proper handling of input boundaries, or real-time data processing speed.
- *Dependability Attributes:* Built-in error logging, memory management, or predictable failure states (fail-safe).

Select and Justify the Acceptance Method

The acceptance method should be selected based on the characteristic being verified, the availability of vendor evidence, and the risk associated with the software's intended use. For software, organizations commonly combine direct testing with vendor-process evaluation so that both performance in the user's environment and dependability of the supplier's development controls are addressed.

- **Method 1: Special Test / Inspection:** The organization verifies selected characteristics by running controlled tests, inspections, analyses, or benchmark problems with known or independently verified results. This method is especially useful for demonstrating calculation accuracy, boundary behavior, installation integrity, and performance within the approved operating environment.

- **Method 2: Commercial Grade Survey:** The organization evaluates the supplier's relevant controls, such as release management, defect tracking, regression testing, corrective action, and customer notification. Survey evidence can support dependability-related characteristics when the surveyed controls are directly applicable to the software version and intended use.
- **Method 3: Source Verification:** The dedicating entity directly verifies selected activities or outputs at the source. For commercial software, this method is less common because proprietary development information may not be available.
- **Method 4: Acceptable Supplier or Item Performance Record:** The organization uses documented operating history or supplier performance information when the record is relevant, traceable, and sufficient to support the specific critical characteristic being accepted.

Document and Approve the Dedication Package

The final dedication package should bring together the technical evaluation, critical characteristics, acceptance criteria, test or survey results, deviations, justifications, and approvals. Approval should be limited to the specific software version, configuration, operating environment, users or use conditions, and technical scope evaluated during dedication.

Typical Quality Records and Evidence

- Software inventory entry, including owner, version, configuration, and approved use.
- Classification basis and safety significance evaluation.
- Software QA plan or implementing procedure requirements.
- Requirements, intended-use statement, and operating environment definition.
- Verification and validation plans, test cases, test results, deviations, and approvals.
- Commercial grade dedication plan, technical evaluation, critical characteristics, acceptance methods, and dedication report.
- Vendor survey report, supplier corrective action information, and defect-notification agreements, where applicable.
- Configuration baseline, approved users, access controls, backup evidence, and change history.

- Impact reviews and regression testing records for patches, upgrades, or scope changes.

Critical Software CGD Constraints

- **Version Locking:** Dedication applies to a *specific version* of the software, such as Version 2026.1. If the vendor issues an automated software patch or upgrade, such as Version 2026.2, the prior dedication may no longer be valid until the change is evaluated and any required regression testing is completed before safety-related or safety-affecting use.
- **Bounded Dedication:** Software is dedicated only for the exact parameters tested. If a calculation tool is dedicated for linear stress analysis, engineers should not use it for *non-linear* safety calculations until the dedication package is expanded with new non-linear test problems.

Combining CGD Methods

Using **Method 1** and **Method 2** together often provides a balanced dedication strategy for commercial software. Method 1 confirms that the software performs as expected in the purchaser's defined application, while Method 2 evaluates whether the supplier's development, release, and defect-control processes support continued dependability.

This combined approach is useful because software risk is not limited to calculation output. The organization also needs confidence that released versions are controlled, known defects are tracked, and future changes or error notices can be evaluated before continued safety-related or safety-affecting use.

Step 1: Execute Method 2—Commercial Grade Survey

Before relying on vendor-provided evidence, the QA team should determine whether a commercial grade survey is appropriate and define the exact process areas to be evaluated. Because proprietary source code is often unavailable, the survey should focus on controls that affect the released software product and the organization's ability to respond to future defects or changes.

Critical Attributes to Audit:

- **Configuration Management:** Does the supplier control source code baselines, restrict code changes through controlled access, require peer review, and authorize releases through a formal release process?
- **Testing and Integration:** Do they perform automated regression testing before releasing a new software patch or version?
- **Error Reporting & Notification:** Does the supplier maintain a formal system to track software defects and provide a documented notification process for software defects, patches, version changes, and safety-significant errors?

Practical Output:

If the survey is successful, the organization documents a **Commercial Grade Survey Report**. This report allows the organization to accept the vendor's internal verification data, satisfying "dependability" critical characteristics, such as code stability and built-in error handling, without having to reverse-engineer the software.

Step 2: Execute Method 1—Special Test/Inspection

After vendor-process evidence has been evaluated, the organization should verify the software directly in the environment where it will be used. Method 1 testing should be tied to the approved scope of use and should demonstrate that the software satisfies the functional characteristics credited in the dedication package.

Practical Implementation:

- **The Benchmark Matrix:** Create a suite of "Problem Sets" or test cases with known, verifiable answers. These answers must be derived from trusted sources, such as textbook analytical formulas, hand-calculations, or physical laboratory data.
- **Boundary and Error Testing:** Input data at the edge of acceptable limits, such as maximum processing speeds, extreme temperature limits, or complex geometric shapes. Verify that the software processes the boundaries correctly or generates the expected error flag.

- **Environment Verification:** Run these tests on the exact operating system, hardware, and network configuration that engineers or operators will use in production.

Step 3: Integrate Both Methods

The following process map shows how the two methods work together from planning through final approval.

Process Step	Method 2: Vendor Survey	Method 1: Special Test / Inspection	Dedication Output
1. Plan the dedication	Identify vendor process areas to evaluate, including configuration management, release control, regression testing, and defect notification.	Identify the benchmark tests, acceptance tolerances, operating environment, and functional boundaries for intended use.	Approved dedication plan that assigns each critical characteristic to the appropriate acceptance method.
2. Collect evidence	Perform the commercial grade survey and document vendor controls, quality records, release practices, and corrective action processes.	Run benchmark, boundary, installation, and functional tests on the same version and environment intended for production use.	Survey report, test procedure, completed test records, deviations, and objective evidence.
3. Evaluate results	Confirm whether vendor processes support dependability characteristics, such as release integrity, defect tracking, and notification controls.	Confirm whether software performance meets the required functional characteristics, accuracy limits, and bounded use conditions.	Technical evaluation showing which characteristics passed, which require compensatory controls, and which are outside the approved scope.
4. Approve and control use	Retain vendor survey results and establish the error-notification loop for future defects, patches, and version changes.	Lock the accepted software version, hardware or operating environment, benchmark baseline, and approved scope of use.	Approved dedication package for the specific software version, configuration, environment, and intended use.

Step 4: Map Characteristics to Acceptance Methods

In the dedication plan, explicitly declare which method satisfies which characteristic:

- *Mathematical accuracy of solvers:* **Method 1** (tested in-house).
- *Adequate software regression testing:* **Method 2** (verified via vendor audit).
- *Prompt defect notification:* **Method 2** (verified via vendor audit).

Step 5: Establish the Vendor Error-Notification Loop

The Method 2 survey should establish a documented notification process with the vendor for software defects, patches, version changes, and safety-significant errors. The internal procedure should require that any vendor notification automatically triggers an engineering review to determine whether existing nuclear safety designs, calculations, or logging configurations are impacted.

Step 6: Maintain the Dedicated Configuration

When the vendor releases a patch or version change, the organization should evaluate the change before continued safety-related or safety-affecting use.

- Review the **Method 2** data provided by the vendor regarding what changed in the patch.
- Re-run a targeted subset of the **Method 1** benchmark or regression tests to confirm the patch did not affect credited safety functions.

Beginner's takeaway: Keep one clear trail from the software's purpose to its risk classification, required controls, test or acceptance evidence, approved version, limits of use, and change history. If any link is missing, stop and resolve it before relying on the software.

Conclusion

A successful NQA-1 software program applies controls in proportion to software function, safety significance, and failure consequence. High-rigor custom software may require full life-cycle control, independent verification and validation, strict configuration management, and formal change approval. Commercial design and analysis tools may require bounded dedication, benchmark testing, vendor-process evaluation, and version-specific acceptance. Lower-risk tools may still require identification, access control, backup protection, and evidence of fitness for use.

Traceability is the central discipline: each software item should connect to its intended function, safety classification, applicable requirements, verification evidence, approved configuration, and change history. When these elements are maintained, the graded approach becomes a defensible quality tool that protects safety, supports engineering efficiency, and provides objective evidence for regulators, customers, and internal reviewers.

Acronyms and Key Definitions

Use this section as a quick reference while reading the article. The definitions are written in plain language for beginners; applicable standards, regulations, contracts, and approved procedures govern formal use.

Acronym or Term	Plain-Language Definition
ACL — Access Control List	A list of users or groups that are allowed to view, use, or change a software system or its data.
Approved Baseline	The formally accepted software version, configuration, operating environment, documentation, and scope of use that must be maintained under change control.
ASME — American Society of Mechanical Engineers	The standards organization that publishes ASME NQA-1.
Black-Box Validation	Testing software by comparing its outputs with known or independently verified answers without reviewing or changing the source code.
Bounded Dedication	Approval that applies only to the specific software version, configuration, environment, functions, parameters, and limits evaluated during dedication.
CAD — Computer-Aided Design	Software used to create, modify, or review engineering drawings and models.
CGD — Commercial Grade Dedication	A documented acceptance process used to provide reasonable assurance that a commercial-grade item, service, or software product will perform its intended safety function.
CM — Configuration Management	The process for identifying and controlling approved software versions, settings, documents, changes, and operating environments.
COTS — Commercial Off-the-Shelf	A commercially available software product developed for the general market rather than specifically under the purchaser's nuclear quality assurance program.

CRB — Change Review Board	A designated group that reviews, approves, and tracks proposed changes to controlled software or its configuration.
Critical Characteristic	A feature or capability that must be correct for the software to perform its approved safety function.
DOE — U.S. Department of Energy	The federal department responsible for national energy policy and many nuclear facilities, programs, and quality requirements.
FEA — Finite Element Analysis	A computer-based engineering method used to predict stress, deformation, temperature, flow, or other physical behavior.
FRS — Functional Requirements Specification	A controlled document that states what the software must do and the conditions, limits, and interfaces that apply.
HIL — Hardware-in-the-Loop	A test method that connects software to simulated or actual hardware so its behavior can be evaluated under realistic conditions.
I&C — Instrumentation and Control	Systems that monitor plant conditions and control equipment or processes.
IQ — Installation Qualification	Documented evidence that software and supporting components were installed correctly in the approved environment.
Life Cycle	The stages through which software passes, such as requirements, design, implementation, testing, operation, maintenance, and retirement.
NQA-1 — Nuclear Quality Assurance-1	The ASME standard that establishes quality assurance requirements and guidance for nuclear facility applications.
NRC — U.S. Nuclear Regulatory Commission	The federal agency that regulates civilian uses of nuclear materials and nuclear power.
Objective Evidence	Records, test results, observations, reports, or other verifiable information showing that a requirement was met.
OQ — Operational Qualification	Documented evidence that installed software operates as intended across its approved functions and conditions.

PLC — Programmable Logic Controller	An industrial digital controller programmed to monitor inputs and control equipment or processes.
QA — Quality Assurance	Planned and systematic activities used to provide confidence that work, items, services, and software meet applicable requirements.
QAL — Quality Assurance Level	An organization-specific label used to indicate the level of quality controls assigned according to safety significance or risk.
Regression Testing	Repeating selected tests after a change to confirm that previously accepted functions still work correctly.
RPS — Reactor Protection System	A system that detects unsafe reactor conditions and initiates protective actions.
SCADA — Supervisory Control and Data Acquisition	A system used to monitor equipment, collect operating data, and sometimes provide supervisory control.
SIL — Software Inventory Log	A controlled list of software items that records information such as owner, version, configuration, purpose, location, classification, and approval status.
SSC — Structure, System, or Component	A physical plant element that may perform or support a safety or operational function.
V&V — Verification and Validation	Verification checks that software work products meet specified requirements; validation checks that the completed software performs correctly for its intended use.

References

- American Society of Mechanical Engineers. (2015). *ASME NQA-1-2015: Quality assurance requirements for nuclear facility applications*. ASME.
- Moreau, N. P. (2024). *Software quality assurance: An ASME NQA-1 perspective of software used in design and analysis of nuclear facilities*. National Reactor Innovation Center, Idaho National Laboratory.
- U.S. Department of Energy. (2011). *Guidance for commercial grade dedication*. Office of Environmental Safety and Quality.
- U.S. Department of Energy. (2019). *DOE-HDBK-1230-2019: Commercial grade dedication application handbook*.
- U.S. Nuclear Regulatory Commission. (2024). *Regulatory Guide 1.164, Revision 1: Dedication of commercial-grade items for use in nuclear power plants*.
- U.S. Nuclear Regulatory Commission. (2026). *10 C.F.R. § 21.3: Definitions*. Electronic Code of Federal Regulations.
- U.S. Nuclear Regulatory Commission. (2026). *10 C.F.R. Part 21: Reporting of defects and noncompliance*. Electronic Code of Federal Regulations.

Author Information

Laura Suddath is an NQA-1 subject matter expert and quality assurance professional experienced in nuclear quality programs, supplier evaluation and audit, Commercial Grade Dedication guidance, technical documentation, and implementation of quality requirements for safety-related work. She is currently a Quality Engineer, Lead Auditor (AS9100 and NQA-1), and Certified Welding Inspector focused on translating nuclear regulatory and code requirements into practical, auditable controls.

Disclaimer

This article reflects the author's professional opinion and is provided for general informational purposes only. It does not establish company, contractual, technical, regulatory, or quality assurance requirements; assign RISC categories; authorize reduced treatment; or replace applicable customer, contract, licensing-basis, regulatory, or approved quality program requirements.

For information or questions about this article, please contact Laura Suddath at laurasuddath@gmail.com.