

Fermilab Software Quality Assurance Program

v 1.0

October 25, 2013

Approved By: 
T.J. Sarlina, Quality Assurance Manager

Approved By: 
Victoria White, Chief Information Officer

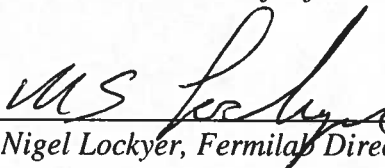
Approved By: 
Nigel Lockyer, Fermilab Director

Table of Contents

I. PURPOSE, SCOPE, AND EXCLUSIONS.....	3
II. ROLES & RESPONSIBILITIES	4
III. GRADED APPROACH TO SQA	5
IV. SQA PROGRAM ELEMENTS.....	6
4.1 Lifecycle Activities.....	6
4.1.1 Requirements	6
4.1.2 Design.....	7
4.1.3 Development.....	7
4.1.4 Verification and Validation	8
4.1.5 Retirement.....	8
4.1.6 Review/Analysis.....	8
4.2 Procurement.....	8
4.3 Change/Release Management.....	8
4.4 Problem Reporting & Corrective Action	9
4.5 Tools, Techniques, Methods, Standards, Practices and Conventions	9
4.6 Documentation.....	10
4.6.1 Quality Assurance Documentation	10
4.6.2 Software Requirements Documentation.....	10
4.6.3 Software Design and Implementation Documentation	10
4.6.4 Software Inspection and Testing Documentation	10
4.6.5 User Documentation.....	11
4.7 Training.....	11
4.8 Assessments.....	12
4.8.1 Management Assessment	12
4.8.2 Independent Assessment	12
APPENDIX A - SOFTWARE QUALITY ASSURANCE LEVELS	13
APPENDIX B - SOFTWARE QUALITY CONTROL MEASURES.....	14
APPENDIX C - GLOSSARY.....	15
APPENDIX D - ACRONYMS	16
APPENDIX E - REFERENCES.....	16

I. PURPOSE, SCOPE, AND EXCLUSIONS

Purpose:

The Fermilab Software Quality Assurance (SQA) Program defines the minimum quality assurance requirements for software used by Fermilab to achieve its scientific program and mission. Through a graded approach, this program ensures development, management, and delivery of reliable software applications that meet or exceed established requirements and expectations through adequate planning, performing, assessing and improving.

Software quality assurance is implemented using a graded approach based on the analysis of potential risks should the software not perform as intended. Evaluating each software application against potential consequences allows for the application of appropriate quality assurance measures and controls.

Fermilab organizations that develop, purchase, and/or maintain software applications are responsible for following the requirements outlined in the Fermilab SQA Program. All activities requiring a graded approach shall be properly documented by the responsible organization. The responsible organization shall also ensure each application has the appropriate level of procedures, controls, and measures in place to assure the software application performs as required.

Management and oversight of the Fermilab SQA Program is provided through the IT Governance function of the Information Technology Management System, which is a component of the Fermilab Contractor Assurance System.

Scope:

Software is an all-encompassing term that describes all of the non-hardware items associated with the operation and use of a computing system. Software includes operating systems, programming languages, spreadsheets, word processors, databases, and digital media files.

Applications are a form of software developed or configured to perform a specific task or range of tasks. At Fermilab, applications directly support the execution of Laboratory functions, processes, and/or procedures. They include custom and packaged programs that provide functions such as performing scientific data analysis, calculating shielding statistics, or tracking safety related information. While many tools such as spreadsheets, databases or programming languages are not applications, some solutions developed using these tools may be considered applications.

The Fermilab SQA Program applies to all software applications used at Fermilab to support the laboratory's scientific program and mission. Fermilab's SQA Program is focused on ensuring that these software applications have the appropriate quality control measures in place to ensure that the applications perform as intended against established requirements.

The following factors should be considered when determining whether a particular piece of software is also an application that falls under the scope of the Fermilab SQA Program:

- Provides an important Laboratory capability or functionality

- Is part of a business process
- Its mission criticality or impact can be identified
- Is subject to external compliance reviews (e.g., audits, regulatory reviews, etc.).

Exclusions and Special Provisions:

As described in the Fermilab Integrated Quality Assurance Program, Fermilab does not employ safety software under the definition of safety software in DOE O 414.1D Quality Assurance.

Non-application type software programs and computer configurations designed to operate experiments, tests, accelerator components and associated equipment are excluded from this SQA Program. Examples include PLC logic, Field Programmable Gate Arrays and embedded software. These types of systems are covered in the Fermilab Engineering Manual under Section 4 – System Design, subsection titled Software.

Quality assurance for software and/or applications developed to support the generation of scientific results is addressed using approaches described in ANSI/ASQ Z 1.13-1999, Quality Guidelines for Research. The ANSI standard acknowledges peer review as a primary mechanism for assuring quality in science and encourages the application of sound engineering/scientific principles to the design of supporting computer software to the extent that the risk associated with the scientific research program warrants.

Special provisions for the development, maintenance, procurement, and usage of software products related to environment, safety, and health are specified in Fermilab ES&H Manual Chapter 5201 found at: <http://esh-docdb.fnal.gov/cgi-bin/ShowDocument?docid=394>

II. ROLES & RESPONSIBILITIES

The Information Technology Management System Owner is responsible for overseeing the Fermilab SQA Program, including the development, implementation, and maintenance of the SQA Program in accordance with the requirements in the Fermilab Integrated Quality Assurance Program. The SQA Program will be reviewed for accuracy, relevance, and effectiveness on a one-year cycle.

Division/Section/Center (D/S/C) Heads are responsible for ensuring their organizations are following the SQA program outlined within this document.

Application Owners and/or line managers are responsible for implementing SQA processes and procedures for their software applications that follows the SQA Program explained within this document. They are also responsible for ensuring that proper processes and procedures required by the SQA Program are followed and can show compliance to the SQA program through self-assessment results.

Application Owners are responsible for maintaining an inventory of software applications under their control, assigning appropriate quality assurance levels, and implementing appropriate quality control measures to ensure that performance requirements are met.

III. GRADED APPROACH TO SQA

The graded approach to software quality assurance is based on a scheme used for managing software applications throughout their lifecycle and takes into consideration all of the following factors:

- The relative importance to safety, safeguards and security
- Criticality of software functionality and results produced
- Operational impact of failure
- The magnitude of any hazard involved
- The software application's intended function
- Expected life of the software application
- Dependencies and impact on other software applications or systems

To ensure software applications are adequately categorized and characterized, a single grading system is used to determine the appropriate level of quality assurance rigor that should be applied to each software application. Quality assurance levels are applicable to all software classes and are assigned to specific applications based on the severity of potential consequences that could result if an application does not perform as intended. Reasonable judgment should be used by Application Owners when assigning quality assurance levels.

The following flowchart details the process for determining appropriate quality assurance levels and quality control measures for software applications. Software quality assurance levels are explained in detail in Appendix A.

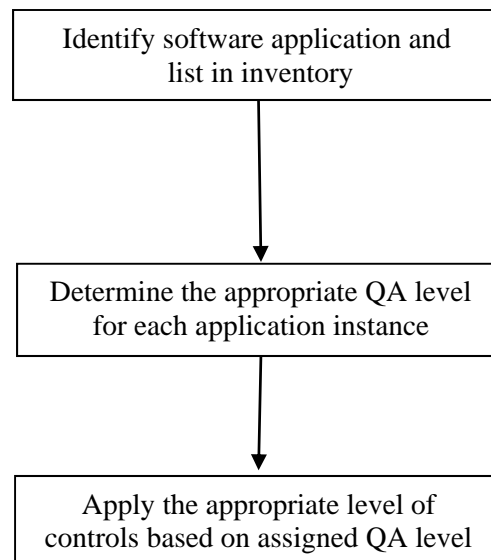


Figure 1. Process for determining the appropriate quality assurance level for specific software applications.

After quality assurance levels are assigned, an appropriate level of controls needs to be applied to each application. The Software Quality Control Measures list defined in Appendix B provides guidelines to follow when applying controls based on the chosen quality assurance level. Following the graded approach, more stringent requirements apply to the “High” quality assurance level than apply to “Moderate” and “Low” QA levels. If an Application Owner, using their discretion and judgment, decide to tailor the degree to which the mandatory controls are applied, the tailoring shall be formally documented.

IV. SQA PROGRAM ELEMENTS

4.1 Lifecycle Activities

An important aspect of software quality is the software lifecycle. Listed here are the most important activities that need to be considered when choosing appropriate practices and selecting a software process. Application Owners are expected to adapt practices and processes from these standards that are appropriate for the assigned software quality assurance level. Applications that are not classified with a High Quality Assurance Level can choose a reduced or simplified process in each of the areas.

Existing standards cover lifecycle processes and activities in detail. An applicable standard is IEEE ISO/IEC 12207:2008, containing common practices and specific recommendations for software lifecycle processes. In addition, IEEE 730 contains information about forming software quality assurance plans, and IEEE 1012 defines software verification and validation processes.

4.1.1 Requirements

Documenting stakeholder and system requirements is a known way to properly focus and guide application development. The recommended practice is to trace all features and functionality developed for an application through system requirements up to stakeholder requirements. Requirements also guide system design and production of test cases. Test cases validate requirements, therefore providing excellent user acceptance criteria. IEEE 830 provides an excellent description and template for high-level requirement documenting. Requirements describe the functions that a system must do or how well the system must perform a function.

In collaboration with key stakeholders, Application Owners are responsible for defining and documenting design requirements at an appropriate level to ensure that the software application meets expectations. Requirements should be traceable throughout the software development lifecycle.

4.1.2 Design

The design activity is the process of problem-solving and planning for a software solution. Design documents describe how a feature will look, how it will be developed, and how it will be integrated into the full system. The level of detail in design documents must be commensurate with the quality assurance level. Designs must be tied to system and stakeholder requirements. Without this tie, the feature being designed cannot be justified. Design documents provide an excellent source for early review before further development begins.

Application Owners are responsible for documenting software designs at a level commensurate with the designated quality assurance level. Commercially purchased software applications do not require design documentation. However, design documentation is required when commercial software applications are modified to create specific applications.

Application Owners are responsible for organizing and conducting formal design reviews at a level commensurate with the designated quality assurance level. The results of design reviews should be documented and corrective actions should be identified and tracked to completion as appropriate.

4.1.3 Development

The development is the process of writing and maintaining the source code, but in a broader sense of the term, it includes all that is involved between the conception of the desired software through to the final build of the software, ideally in a planned and structured process. It consists of developing the design output (e.g., source code) using a programming language, application interface, or other form of development tool suitable for compilation or translation into an executable code. The design, as described in the software design description, is used as the basis for the application development and may need to be modified to reflect changes identified during the implementation phase.

Software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software product. It also may include integration of multiple artifacts into a whole system as well as testing of the changes or newly developed programs.

Application Owners are responsible for determining an appropriate development methodology to ensure that design requirements are successfully met.

4.1.4 Verification and Validation

Verification is the process of evaluating the work-products of a development phase to determine whether they meet the specified requirements.

Verification activities include review, walkthroughs and inspections of requirements, design, code and test cases. Validation is the process of evaluating software during or at the end of the development process to determine whether it satisfies the requirements. Validation is completed via testing the software product.

Installation and acceptance activities shall be appropriately planned, executed and documented. Applications with high or moderate QA levels should have a documented test plan, including test cases, and test results. Additionally, for applications assigned a high QA level, all issues identified during acceptance testing must be documented and the disposition of each issue must be tracked.

4.1.5 Retirement

When an application is no longer needed, the Application Owner is responsible for removing it from service, updating its state in the application inventory, and providing appropriate notification to the application user community.

4.1.6 Review/Analysis

Application Owners should conduct regular assessments throughout the software lifecycle for applications that have been assigned a high QA level. Assessments should be conducted on an appropriate level for applications assigned moderate or low QA levels.

4.2 Procurement

Software acquisition and software services should be procured in a controlled manner to ensure conformance with specified requirements and expectations of the purchaser. Acquisition documents for commercial software should clearly state or reference requirements for the software being acquired. Acquisition documents and Statement of Work for contracted software development and software services should identify all documentation, plans, and procedures to be supplied by the vendor. This vendor supplied documentation for software services/code development must be based on the QA level of the software and follow the mandatory/recommended documentation as outlined in Appendix B.

4.3 Change/Release Management

During the application lifecycle, software systems and applications typically experience a number of changes as new features are added and issues are addressed. Change Management is necessary to ensure that changes are evaluated and controlled in a manner commensurate with the software QA level. In particular, Application Owners shall develop, specify, or identify Change Management processes to ensure

that changes are recorded, evaluated, prioritized, authorized, planned, tested, and implemented in a manner that is commensurate with the QA level of the application.

The Change Management process should establish how change requests are generated, reviewed and approved, as well as acceptance criteria for the end result. The Change Management process should also define the authority for determining changes in distribution modules such as releases, service packs or patches. The required documentation at each stage of the Change Management process, as well as the designated decision-making mechanisms should be directly commensurate with the software QA level.

The release management process should establish a method for packaging, testing and delivering new or changed components into the production environment with minimal disruption to existing service. A release must be uniquely identified with a version number.

A key component of release management is version control. Version control is necessary to ensure that the proper version of an application or developed component is being used. The required level of software version control and management varies by software QA level, and it is the responsibility of the Application Owner to designate and/or document the chosen methodology and approach. Software version control must be established prior to the application being placed into production use.

Where appropriate, a mechanism should be established to facilitate traceability of resulting code or configuration changes to the initial change request. It is also advantageous to provide traceability with the change request approval, implementation timeline, and evaluation of end results. Establishing such a mechanism reinforces the relationship between change management and configuration management.

4.4 Problem Reporting & Corrective Action

Application Owners are responsible for implementing processes that collect and record problems, and ensure corrective actions are identified and tracked through to completion.

Incident reporting for applications tracked by the Fermilab Service Desk shall use the service desk application to report incidents and track corrective actions through to completion following Fermilab IT Service Management processes.

4.5 Tools, Techniques, Methods, Standards, Practices and Conventions

This SQA Program is intended to provide guidance to Application Owners with the development, implementation, and maintenance of software applications under their control, to ensure that applications perform as intended. Tools, techniques, methods, standards, practices or conventions are not specified in this SQA Program; however;

whenever possible a common set of practices should be established within application areas to ensure consistency and implementation of the SQA program.

4.6 Documentation

4.6.1 Quality Assurance Documentation

Software applications shall be documented to ensure that their purpose, use and acceptance criteria are clearly understood, and to support ongoing maintenance to ensure the continued functionality of the applications. Application Owners are responsible for identifying the documents needed to accomplish these objectives and determining the level of control required. All documentation that requires formalized control shall be controlled in accordance with the Fermilab Document Management & Control Policy.

4.6.2 Software Requirements Documentation

Software requirements documentation should include:

- A description of the functions the application is intended to perform.
- Time-related performance issues (e.g. speed, recovery time, and response time).
- Attributes of software operations (e.g. portability, acceptance criteria, access control, and maintainability)
- External interfaces including interactions with people, hardware, and other software applications.

4.6.3 Software Design and Implementation Documentation

The following elements should be considered when developing software design and implementation documentation:

- Major component descriptions of the software design (as related to requirements)
- Technical description
- Description of the prescribed ranges for inputs and outputs. For applications assigned “High” quality assurance levels, documentation should include full descriptions of user inputs and outputs.
- Design descriptions that can be translated into code. For applications assigned “High” QA levels, details should be at the code module level. For “Moderate” and “Low” QA levels, high-level descriptions of external interfaces and major logic structure are adequate.
- Source code.

4.6.4 Software Inspection and Testing Documentation

Inspection and acceptance testing of software should include objective evidence of the review of software activities, lifecycle documentation, and test reports to ensure that the software:

- adequately and correctly performs all intended functions;
- meets all established requirements;

- 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 outcomes of the software.

Testing may include unit tests, integration tests, system tests, regressions tests and user acceptance tests. Inspections may range from a set of peers conducting a formal meeting to review documentation, to a single peer performing a desktop check of the documentation or code listing.

Inspection and testing documentation should be developed and organized so that it is traceable to the software requirements and design, and should contain the results of inspection and testing activities. This includes results of reviews and tests, and summary status of the software. For applications with a high QA level, each of the following documents should be created and reviewed during each phase of software testing: (1) Test Plan, (2) Test cases, and (3) Test Results. For software at the lower QA levels (moderate/low), the test plan, test cases and test results may be combined into a single document that is reviewed after completion of acceptance testing.

4.6.5 User Documentation

User documentation should include the following:

- User instructions that include an overview of the application's purpose and functionality, description of the user's interaction, and description of any required training
- Input and output specifications
- Input and output formats, where appropriate
- Description of system limitations

The type and detail of user documentation will vary depending on software class, function, and quality assurance level assigned. For lower QA levels, documentation should consist of basic operations and an explanation of common errors. For all cases, user documentation may be in the form of online help.

4.7 Training

All personnel associated with the development, acquisition, configuration, management, use, oversight, and retirement of Fermilab software applications shall have the appropriate training necessary to perform their assigned job per the Integrated Quality Assurance Program, either through experience, classroom, or on-the-job training.

Fermilab line management is responsible for establishing job requirements and for ensuring personnel receive the appropriate level of training and continuing training necessary so that job competency and compliance is maintained.

Any employee performing software quality assurance activities, including verification, validation, and auditing should be qualified through experience and/or training.

4.8 Assessments

Implementation of effective quality assurance methods will be verified through Fermilab's Assessment process as described in the Integrated Quality Assurance Program. Nonconformities found during this process shall be recorded and tracked through to closure.

4.8.1 Management Assessment

Management assessments are used by an organization to evaluate its own management processes and their implementation in an effort to identify good and noteworthy practices, uncover issues, identify corrective actions, and ensure that the work being performed is satisfactory and in accordance with requirements. The management assessments shall be designed to verify that established processes and procedures, requirements and grading are sufficient and accurate.

4.8.2 Independent Assessment

Independent assessments can be performed by internal or external resources. A peer review qualifies as an independent assessment. Internal resources will be appropriately qualified and have sufficient authority to allow for an unbiased assessment. Independent assessments can include externally imposed audits and reviews of the SQA program.

APPENDIX A - SOFTWARE QUALITY ASSURANCE LEVELS

Quality Assurance Level	Potential Consequences if Software Does Not Perform as Intended
High (If one or more apply)	Causes injury
	Causes an evacuation
	Causes environmental hazard including medium-high potential for reportable radiological release, groundwater contamination, or regulatory violation
	Causes a significant disruption in laboratory operations or business operations
	Causes significant disruption of an experiment or program, or has significant impact on a contractor or DOE mission or program
	Compromises data integrity: total loss of or severe reduction in data quality, experimental data or equipment output
	Causes a release of DOE sensitive information
	Can lead to compromises in systems with personal identifiable information (PII) or release of passwords or credentials that lead to compromise of such systems
Moderate (If one or more apply)	Causes environmental hazard including low/small potential for reportable radiological release, groundwater contamination, or regulatory violation
	Causes minor program downtime
	Causes a minor disruption in laboratory operations or business operations
	Incurs a minor loss of experimental data or equipment output
	Causes minor disruption to an experiment or program, or has minor impact on a contractor or DOE mission or program
	Causes a loss public release of information not authorized by DOE for public release
	Minor reduction in data quality or equipment output
	Can lead to compromises in systems or the release of passwords or credentials that can lead to compromise of such systems
Low (All should apply)	Does not present cause a worker safety-hazard
	Does not result in any evacuation
	Does not cause environmental hazards, causes no environmental impact
	No program downtime
	No/Negligible disruption in laboratory operations or business operations
	No/Negligible loss of experimental data or equipment output
	No adverse public impact
	Causes loss of information that is authorized by DOE for public release. No unplanned release of information to the public.
Negligible reduction in data quality or equipment output	

APPENDIX B - SOFTWARE QUALITY CONTROL MEASURES

M = Mandatory

R = Recommended

O= Optional

Control Measure	High	Moderate	Low
Change Management practices in place to ensure that changes are recorded, evaluated, prioritized, authorized, planned, tested, and implemented per the process.	M	M	R
Sufficient level of detail in the functional requirements to develop test cases.	M	R	R
Documentation created to design, develop, and maintain the software application.	M	M	R
Software source under version control	M	M	M
All external interfaces identified, documented and analyzed including user inputs and outputs	M	R	R
High-level descriptions of major logic structure	M	R	R
Appropriate Test Plan, Test cases and Test Results documents created (refer to section 4.7.4)	M	M	M
User documentation under version control	M	R	R
Documentation of basic operations and explanations of common errors	M	M	R
Acquisition documents created for procurement of software or software services	M	M	M
Inspections and review documentation created and controlled during each of the lifecycle phases and at the end of the development cycle.	M	R	R
Each issue identified during inspection and acceptance testing, and its disposition, shall be documented.	M	O	O
Results of code reviews and acceptance test results must be reviewed and approved prior to production release.	M	R	O

APPENDIX C - GLOSSARY

Application – A form of software developed or configured to perform a specific task or range of tasks. An application can be a complex system with many components/programs each of which may have a different SQA level. The application should be classified as the highest level of the components, but each component may have different controls based on its individual SQA level. In the context of this document, software such as firmware is considered an application.

Application Owner - The individual or group with the responsibility to ensure that the program or programs which make up the application, accomplish the specified objective or set of user requirements established for that application, including appropriate security safeguards.

DOE Sensitive Information - Sensitive unclassified data, such as personally identifiable information (PII), official use only, and unclassified controlled nuclear information require special handling and protection to prevent misuse of the information for inappropriate purposes. (from <http://energy.gov/ig/downloads/audit-report-ig-0818>)

Loss of Availability - disruption of access to or use of information or an information system.

Loss of Confidentiality - unauthorized disclosure of information.

Loss of Integrity - unauthorized modification or destruction of information.

Software – All of the non-hardware items associated with the operation and use of a computing system.

Software Quality Assurance – Processes and procedures that ensure that a software application meets or exceeds established requirements.

APPENDIX D - ACRONYMS

Acronym	Description
IQA	Integrated Quality Assurance
ITIL	Information Technology Infrastructure Library
SQA	Software Quality Assurance
FIPS	Federal Information Processing Standards

APPENDIX E - REFERENCES

- Fermilab Engineering Manual (location: http://www.fnal.gov/directorate/documents/FNAL_Engineering_Manual.pdf)
- SQASG-TP-10-01 Rev 1 Non-Safety SQA (location: <https://sharepoint.fnal.gov/cd/sites/opm/QA/SoftwareQA/Shared%20Documents/Reference%20Documents/SQASG-TP-10-01%20Rev%201%20Non-Safety%20SQA.pdf>)
- FEDERAL INFORMATION PROCESSING STANDARDS - Standards for Security Categorization of Federal Information and Information Systems (location: <http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf>)
- CHANGE MANAGEMENT ITIL PROCESS – ITIL documentation can be found in DocDB (location: <https://cd-docdb.fnal.gov:440/cgi-bin/ListBy?topicid=187>)