SANDIA REPORT

Sandia National Laboratories Advanced Simulation and Computing (ASC) Software Quality Plan Part 1: ASC Software Quality Engineering Practices Version 1.0

Edward A. Boucheron, Richard R. Drake, H. Carter Edwards, Molly A. Ellis, Christi A. Forsythe, Robert Heaphy, Ann L. Hodges, Constantine Pavlakos, Joseph R. Schofield, Judy E. Sturtevant and C. Michael Williamson

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Sandia National Laboratories Advanced Simulation and Computing (ASC) Software Quality Plan

Part 1: ASC Software Quality Engineering Practices

Version 1.0

Edward A. Boucheron, Thermal/Fluid Computational Engineering Sciences
Richard R. Drake, Computational Physics Research and Development
H. Carter Edwards, Advanced Computational Mechanics Architectures
Molly A. Ellis and Christi A. Forsythe, Enabling Technologies
Robert Heaphy, Discrete Algorithms and Math
Ann L. Hodges, SEPR Systems Engineering
Constantine J. Pavlakos and Judy E. Sturtevant, Visualization and Data
Joseph R. Schofield, Capability Development
C. Michael Williamson, Software Systems

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0826

Abstract

The purpose of the Sandia National Laboratories Advanced Simulation and Computing (ASC) Software Quality Plan is to clearly identify the practices that are the basis for continually improving the quality of ASC software products. The plan defines the ASC program software quality practices and provides mappings of these practices to Sandia Corporate Requirements CPR 1.3.2 and 1.3.6 and to a Department of Energy document, *ASCI Software Quality Engineering: Goals, Principles, and Guidelines*. This document also identifies ASC management and software project teams' responsibilities in implementing the software quality practices and in assessing progress towards achieving their software quality goals.

Acknowledgements

Table of Contents

List of Figures

List of Tables

Foreword

The genesis of this document is a long and involved tale. In the beginning the ASC Advanced Applications program element leader undertook rewriting *ASCI Applications Software Quality Engineering Practices, Version 2*, utilizing the experience gained through two internal and one external NNSA SQE practices assessments. Simultaneously the ASC program office decided to expand the scope of this rewriting effort to encompass the entire range of software development activities within the ASC program at Sandia. The writing team was correspondingly expanded to include representatives from the larger code development community. The writing team's charter was reworked to embrace this new constitution and to establish a revised set of end goals.

The broad and diverse community of code development embodied within the ASC program naturally leads to a wide range of software quality practices and philosophical underpinnings as to what software quality practices should be. This makes the task of finding common ground and coming to agreement on software quality practices particularly difficult.

The best analogy that comes to mind is that of religion, with the various authors of the document representing different faiths; they gather around the table to discuss and write, and it is an uneasy ecumenical council. As relationships within the team mature, all learn to acknowledge each others different beliefs, to respect each others opinions and to drive toward as best a set of compromises as possible. Due to the maturity and professionalism of the individuals involved, an impressive ability to subsume individual desires in favor of realized teamwork can be witnessed. Unfortunately, returning such a committee result to the development communities represented in this writing effort did not meet with similar understanding "around the table." A formal comment resolution process that had been adopted by the writing team received a deluge of substantive comments, thus providing ample evidence through sheer volume of comments alone that the intended inclusive goals of the document were not being realized.

At this point, management concluded that the comment resolution process was not achieving its intended goal because it did not allow broad or sweeping changes that could accommodate what much of the practitioner community was requesting. Many reviewers commented on the value of individual sections of the draft documents and many statements that it was "very close", led to the idea that a small, tight team, focused in intent, could quickly and easily modify the current draft to conform to their needs and be satisfied with the result. The AQMC heartily agreed and immediately appointed one representative from each program element to form a small team that would spend two days "locked in a room" with the express goal of producing a distilled document. This final review team consisted of the following individuals: Ted Blacker, Edward A. Boucheron (chair), H. Carter Edwards, Molly Ellis, Patricia Hackney (editor), Robert Heaphy, Sue Kelly, Bob Kerr, and Judy Sturtevant.

In the end, after all is said and done, one must leave the table and practice whatever faith you believe you are committed to. As chair of the final review team, I accept full responsibility for the particular SQE faith represented in this document, realizing full well that it may be viewed as heresy by some.

Edward A. Boucheron, Chair

Executive Summary

The purpose of the Sandia National Laboratories (SNL) Advanced Simulation and Computing (ASC) Software Quality Plan is to clearly identify the practices that are the basis for continually improving the *quality* of ASC software products. Quality is defined in DOE/AL Quality Criteria (QC-1) as *conformance to customer requirements and expectations.*

This quality plan defines the ASC program software quality practices and provides mappings of these practices to the SNL Corporate Process Requirements (CPR 1.3.2 and CPR 1.3.6) and the Department of Energy (DOE) document, *ASCI Software Quality Engineering: Goals, Principles, and Guidelines* (GP&G). This quality plan identifies ASC management and software project teams' responsibilities for cost-effective software engineering quality practices.

The SNL ASC Software Quality Plan establishes the signatories' commitment to improving software products by applying cost-effective software engineering quality practices. This document explains the project teams' opportunities for tailoring and implementing the practices; enumerates the practices that compose the development of SNL ASC's software products; and includes a sample assessment checklist that was developed based upon the practices in this document.

Commitment
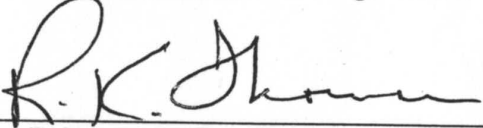
The SNL ASC programs that develop and/or deploy software will follow the practices, processes, and activities outlined in the SNL ASC Software Quality Plan (Software Quality Plan). Our purpose is to produce quality software products that satisfy our customers' requirements and expectations and provide tangible evidence demonstrating high confidence in ASC software projects at SNL. An additional intent of the Software Quality Plan is to foster organizational consistency by defining common practices and by facilitating the use of cost-effective, common tools and processes where feasible. This Software Quality Plan will be modified and improved as the code development process matures.

Approved By:
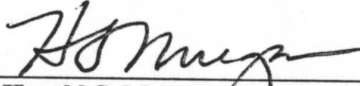
_____   11/18/2004
Michael O. Vahle, ASC Program Director   Date
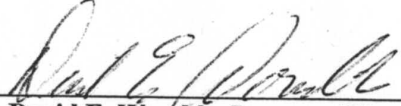
_____   11/23/2004
Robert K. Thomas, ASC Campaign Manager   Date
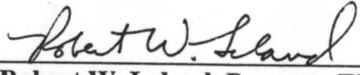
_____   11/23/04
Harold S. Morgan, Program Element Lead   Date
ASC Advanced Applications & Modeling, and
Materials and Physics Models
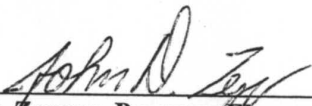
_____   12/15/04
David E. Womble, Program Element Lead   Date
ASC Algorithms and Enabling Technologies

_____   11/23/04
Robert W. Leland, Program Element Lead   Date
ASC Data and Visualization Sciences
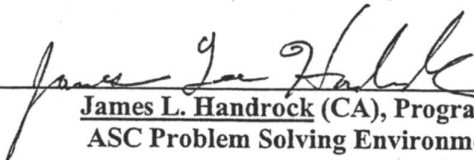
_____   11/23/04
John D. Zepper, Program Element Lead   Date
ASC Physical Infrastructure & Platforms
Computational System & Simulation
Support Program Element Lead

Approved By (continued):
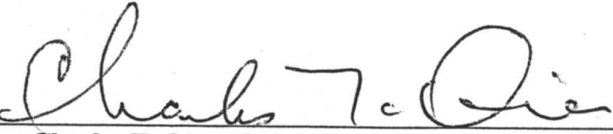
James L. Handrock (CA), Program Element Lead
ASC Problem Solving Environment

12/8/04
Date

John A. Larson (NM), Program Element Lead
Simulation & Computer Science
ASC DisCom/NM Program Element Lead
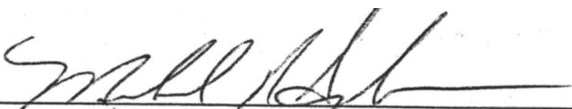
11/23/04
Date

Charles T. Oien (CA), Program Element Lead
Simulation & Computer Science
ASC DisCom/CA Program Element Lead
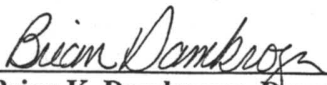
12/8/04
Date

Concurred By:

Michael R. Sjulin, Deputy Director
Stockpile Systems I, SNL/NM
– or –
Patrick A. Sena, Manager
Use Control Systems Department 02115

12/03/04
Date

Brian K. Damkroger, Deputy Director
Stockpile Systems, SNL/CA

12/9/04
Date

This page left blank intentionally

1   Introduction

The National Nuclear Security Agency (NNSA) oversees the Stockpile Stewardship Program (SSP) to provide and ensure confidence in the safety, performance, and reliability of the U.S. nuclear stockpile in the absence of underground testing. To this end, NNSA enabled the Accelerated Strategic Computing Initiative (ASCI) to support the SSP in transitioning from using test-based methods to using more computational and simulation-based methods. Since Accelerated Strategic Computing is no longer an initiative, the program has been renamed Advanced Simulation and Computing (ASC).

The ASC program involves coordination among the three nuclear weapon laboratories, all of which have contributed to the development of a set of guiding principles. The *ASCI Software Quality Engineering: Goals, Principles, and Guidelines* (GP&G) provides direction for all ASC software projects. The GP&G specifies that each laboratory select and tailor their best practices to achieve the stated goals of (1) establishing confidence in codes and (2) establishing credibility in results.

The Sandia National Laboratories *Advanced Simulation and Computing (ASC) Software Quality Plan* (Software Quality Plan) follows ASC program direction from the GP&G. This Software Quality Plan consists of *Part 1: ASC Software Quality Engineering Practices* (Part 1) and *Part 2: Mappings for the ASC Software Quality Engineering Practices* (Part 2). The Software Quality Plan is SNL's implementation of the GP&G and is intended for a broad audience. This document provides the background, high level information and overall practices that the ASC software projects are required to address and is expected to be utilized by the software project team practitioners. Figure 1 illustrates the relationship among the software quality plan, drivers for this plan, and expected project implementations.

The Software Quality Plan, as part of process improvement, is a consolidation of previously separate efforts by the ASC Applications and the Simulation and Computer Science/Ongoing Computing (S&CS/OC) programs based upon feedback from the assessments, adherence to corporate process requirements, and the desire of ASC management to address elements in other quality frameworks (for example, ISO 9000). The Software Quality Plan is intended to combine the efforts of these groups to create one plan for all ASC software projects. The Software Quality Plan replaces the existing Application and S&CS/OC practice documents.

Although this Software Quality Plan was generated to conform with the SNL corporate and QC-1 revision 9 standards, QC-1 revision 10 and DOE O 414.1C were issued during the writing of this document; therefore, SNL's corporate policies are also currently in transition. As such, conformance with the practices contained in this Software Quality Plan will not guarantee conformance with the evolving SNL corporate quality requirements. Mappings of the practices in this document to QC-1 revision 10 are provided in Part 2 as a guide to assess compliance with these standards in transit. This document will be annually reviewed under the oversight of the ASC Quality Management Council (AQMC) to consider revisions, including those required to incorporate or otherwise address changes in the governing standards.
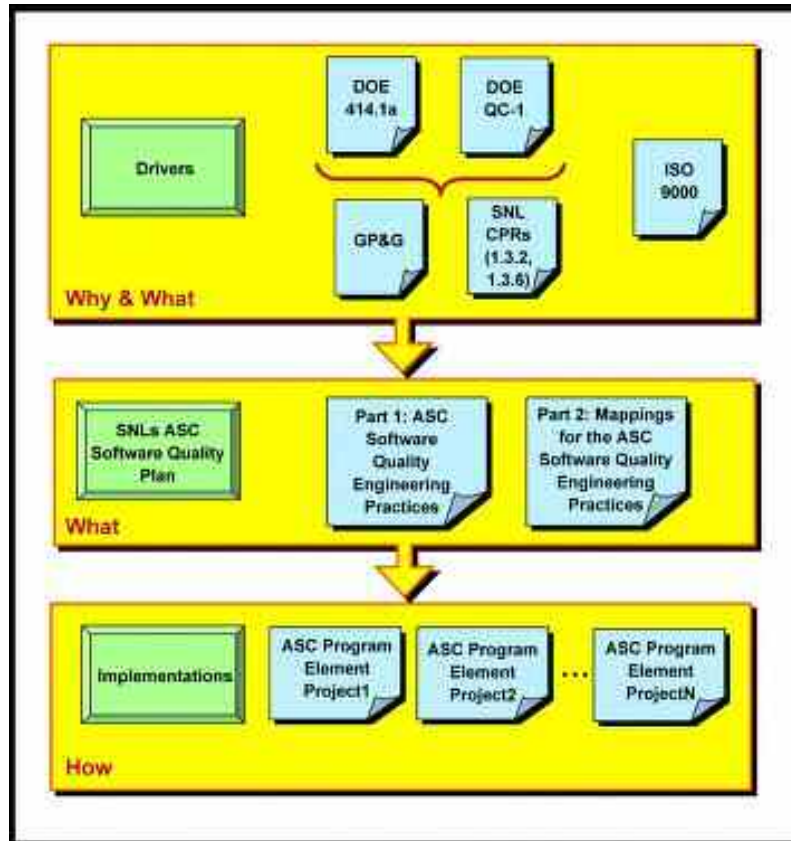
Figure 1. Relationship of Drivers, Software Quality Plan and Project Implementations.

1.1 Quality Definition and Goals

The purpose of this document is to describe software quality engineering practices that lead to a high level of confidence in ASC software products and projects at SNL. The intent of the practices stated herein is to promote *quality* for software products and projects.

Multiple sources for defining quality were studied and a common theme surfaced: not all requirements are explicitly stated, however, all implied as well as explicit needs must be met. Expectations are often defined as customer needs that have not been explicitly stated as requirements. Considering this theme plus the close traceability between the GP&G and the *DOE Quality Criteria* (QC-1), main drivers of this document, the definition from the QC-1 became the basis of the Software Quality Plan:

> *Quality - Conformance to customer requirements and expectations.*

The quality goals of the Software Quality Plan are to:

- provide guidance for software quality engineering practices that will
  - satisfy the stated and implied needs, budget, and schedules of the customer,
  - be effective and cost efficient, and
  - provide a common foundation for ASC projects;
- ensure continual quality improvement of SNL's ASC software products, software operation and support activities, and software development activities; and
- satisfy requirements specified in the ASC GP&G and SNL Corporate Process Requirements (CPR) drivers to the practical extent within the scope of this document.

-13-

1.2 Overview

This document is organized into the following sections:

- Section 1 introduces the Software Quality Plan and provides the goals,
- Section 2 discusses the drivers and standards,
- Section 3 discusses the implementation of the practices,
- Section 4 identifies the Software Quality Engineering (SQE) practices for the ASC software projects, and
- Section 5 discusses the assessment strategy.

2   Drivers and Standards

The Software Quality Plan is based upon the following drivers:

- *ASCI Software Quality Engineering: Goals, Principles, and Guidelines* (GP&G),

- Corporate Process Requirement CPR001.3.2, Corporate Quality Assurance Program, and

- Corporate Process Requirement CPR001.3.6, Corporate Software Quality Assurance.

All requirements specified in these drivers are addressed by mappings in Part 2 of the Software Quality Plan. A mapping of the practices to the GP&G is also included in Appendix G of this document (Part 1). In some cases the mappings identify gaps to various sections or paragraphs contained in the drivers. In many such instances these gaps are handled in other related documents. In other instances, these gaps will be addressed as the Software Quality Plan matures or as SNL and ASC management so direct.

The International Organization for Standardization (ISO) is the source of ISO 9000. The ISO 9000 standard specifies requirements for a quality management system that should address the organizational structure, responsibilities, procedures, processes and resources necessary for implementing quality. The Software Quality Plan, the foundation of the ASC's quality management system, contains these ISO elements. While the ISO 9000 standard is not a primary driver, there is a significant overlap between ISO 9000 and the requirements for the Software Quality Plan. The ASC program is fully aware of ISO 9000 and is interested in identifying gaps that may exist between the requirements as specified in the GP&G, Corporate Process Requirements (CPRs), and the ISO 9000 standard.

The Software Capability Maturity Model® (SW-CMM®) and Capability Maturity Model Integration® (CMMI®) are software capability assessment frameworks developed by the Software Engineering Institute (SEI) to determine a software supplier's capability to deliver a negotiated quality product. Many of the practices in the Software Quality Plan can be mapped directly to the SW-CMM®/CMMI®, although this is not a requirement. The Software Quality Plan attempts to take the most critical software development elements and incorporate them into its own process improvement effort, but no mapping to SW-CMM®/CMMI® is provided.

3   Software Quality Plan Implementation

The Software Quality Plan allows for tailoring of software project activities in implementing the practices. The implementation of the practices described in this Software Quality Plan is the joint responsibility of ASC management and project teams. Stakeholders are expected to provide guidance, concur with the Software Quality Plan, and participate in the implementation details.

3.1   Management Roles and Responsibilities

Management support and advocacy of software quality are required for the successful implementation of this Software Quality Plan. Two distinct management entities are identified: (1) the ASC Quality Management Council (AQMC) and (2) ASC management with oversight or other direct responsibilities

for ASC-funded software projects. Table 1 defines high-level roles and describes associated responsibilities for the AQMC and ASC management.

Table 1. ASC Software Quality Plan Roles and Responsibilities.

Roles	Responsibilities
ASC Quality Management Council	The AQMC is an oversight group that is responsible for: • setting policy and developing strategy for implementing quality systems for all ASC software projects, • sponsoring and promoting the Software Quality Plan and quality initiatives, • ensuring that the Software Quality Plan provides a framework for defining and reviewing quality objectives, • ensuring the Software Quality Plan is communicated and understood by the community, • authorizing modifications to policies and strategies, • reviewing and assessing quality initiatives in the ASC program, • reviewing the results of independent and external assessments, and • convening working groups to support development of policies and strategies.
ASC Management	ASC management, which may consist of several levels of managers, has oversight or other direct responsibilities for ASC-funded software projects. ASC management ensures consistent and cost-effective implementation of the AQMC's policies and strategies and is responsible for: • directing and ensuring project team implementation of this Software Quality Plan that balances risk, quality, cost, and schedule; • maintaining the Software Quality Plan; • approving and tracking the level of formality established for projects under their direction; • monitoring, improving, and documenting compliance with the Software Quality Plan; • sponsoring and determining the scope, goals, and procedure of independent SQE assessments of software projects; • communicating best software quality practices across the ASC software projects; and • identifying organizational and stakeholder training needs and providing necessary training opportunities that map to these organizational needs.

3.2   Stakeholder Expectations

Stakeholders are individuals or organizations, internal and external to SNL, that are actively involved in a project. Customers and users are stakeholders. Stakeholders may not be accountable to the ASC program; therefore, the ASC Software Quality Plan practices cannot be stated for stakeholders. Expectations for stakeholder are provided in Table 2.

Table 2. Stakeholder Expectations.

Role	Expectations
Stakeholder	Project expectations of the stakeholder include: • providing guidance and concurrence with the Software Quality Plan; • identifying, clarifying, and prioritizing their product expectations and requirements; • negotiating acceptance criteria, schedule, and intended use; • participating in appropriate reviews; and • identifying customer support expectations and requirements for the installation, operation, and training of the product.

3.3   Project Team Tailoring and Implementation

The Software Quality Plan provides descriptions and details to the software projects for implementing the practices identified in this document. All software projects are expected to address each of the appropriate practices and are allowed to tailor their implementation. The project team's practices, processes, and artifacts are a natural part of quality software development. These artifacts are the foundation for satisfying customer requirements, obtaining software engineering/quality feedback for continual process improvement, and for demonstrating consistency with the practices.

The Software Quality Plan does not prescribe any specific implementation of these software quality practices. Project team implementation of the practices must take into account the consequences implied if the delivered product fails to meet its intended use(s). The determination of such consequences involves considering the defined mission of the project (see section 4.2.1). Depending upon the identified consequence level and the associated likelihood that the project will not be able to meet its commitments, each ASC software project may tailor implementation of the practices described in this Software Quality Plan. Project tailoring considers risk factors such as software size, complexity, cost, schedule, visibility, and uniqueness as well as project size (see section 4.2.2).

Software products that are identified as supporting a high consequence mission (for example, weapon certification) will need to implement the majority of the practices at a level of formality (LOF) appropriate to the mission consequence. The LOF suggests which practices are necessary and influences how those practices are implemented, reviewed, and approved.

4 ASC SQE Practices

The ASC SQE practices are organized in this document under sections 4.2 Project Management, 4.3 Software Engineering, 4.4 Software Verification, and 4.5 Training. In the GP&G there were three practice areas: Project Management, Software Engineering, and Software Verification (see Figure 2). Each of these GP&G areas contained training. Rather than discuss training three times, this document combines the training into one section for all three areas. This organization responds to the requirement that each site develop specific practices to appropriately implement the guidelines.

Each section first summarizes the overall scope for the area followed by one or more pertinent practice areas (also shown in Figure 2). Each practice area contains a practice table that covers the expectations of ASC management, statements of the practices, and suggested artifacts that demonstrate implementation.

Figure 2. ASC SQE Practice Areas.

4.1 Organization of the Practice Tables

The practice tables contain an overview description, numbered practice statements, numbered artifacts resulting from the practices, example inputs, and example metrics and measurements.

Overview Description

The overview description provides a high level discussion of particular practices that are involved in an area. The overview also provides additional elaboration that is intended to guide the practitioner in implementing the practices described. The overview of one section may reference the overview or details of another related section.

Practices

Practices are software development and deployment activities. Each practice describes the activities and elements that a project team should address in tailoring and implementing the practice for their specific project. Each practice is uniquely numbered in the format **PRx**. Table 6 at the end of section 4 provides a listing of the practices with the artifacts generated by each. Appendix B contains a separate listing of all the practices.

Artifacts

An artifact is a deliverable or work product that is generated as a practice is exercised. Each listed artifact is an example of an output created or modified by the given practice. All appropriate software product artifacts identified by the project team are to be version controlled and change managed as described in section 4.2.2 Configuration Management. Each artifact is uniquely numbered in the format **ARx**. Text in parenthesis following the artifact name helps clarify or explain the artifact. Text in brackets following the artifact name identifies elements included in the artifact. Table 6 at the end of section 4 provides a listing of the practices with the artifacts generated by each. Appendix B contains a separate listing of all the artifacts.

Example Inputs

The inputs suggested are examples of existing resources, information and/or artifacts external to a practice that may be necessary to perform that practice. For example, in section 4.2.5 Risk Management a suggested input, list of subject matter experts knowledgeable about potential risk events, is a resource external to the practice; however, most of the suggested inputs in section 4.3.3 Customer Support are artifacts from other practice areas. A suggested input that is a resource or information external to the practice is identified by a bullet (·) and one that is an artifact from another practice area is identified by that artifact's number (**ARx**). Each suggested input is followed by a parenthetical expression indicating the associated practice.

Example Metrics and Measurements

Metrics and measurements provide quantitative insight into the effective quality of the process and that of the resulting product. In this document a metric is defined as a quantitative measure of the degree to which a system, component, or process possesses a given attribute and a measurement is defined as the dimension, capacity, quantity, or amount of something. Subject matter experts in the final product are involved in specifying metrics designed to increase quality. Strong customer involvement is also recommended. Suggested example process and product metrics are provided for each of the software lifecycle development areas. These are not required but are intended to inspire software project teams to define their own appropriate metrics. Collected process and product metrics form the basis for one of the artifacts, **AR4**, identified in the Process Implementation and Improvement practice table.

In selecting metrics, teams should consider how the metrics will be analyzed (that is, appropriate statistical methods). Metrics should be analyzed and monitored for undesirable "side effects" which are known in the quality world as "unintended consequences."

Note: The words *metric*, *measure*, and *measurement* have limited consensus usage in the software community. The Glossary contains definitions for these terms as used in this document.

4.2   Project Management

Project Management is the systematic approach for balancing the project work to be done, resources required, methods used, procedures to be followed, schedules to be met, and the way that the project is organized. This section begins with the practice table Strategic Planning as a first step in addressing project management followed by the Determination of Applicable Practices and Level of Formality practice table for risk-based assessment. The specific activities identified in the GP&G are then addressed in practice tables under Process Implementation and Improvement, Requirements Engineering, Risk Management, and finally Project Planning, Tracking and Oversight.

4.2.1 Strategic Planning

PROJECT MANAGEMENT Strategic Planning
Overview Description:
An organization defines a project and its mission; management responsibilities and authorities; users and customers; and interelationships with other projects (organizational context). The project's mission is one basis for the selection of appropriate practices. For example, a research project may not need all practices used by a team developing a production product. The organizational context presents an opportunity for organizationally related projects to share common practices, procedures, processes, tools, training, and documentation. Large projects and frameworks may form their own organizational context which allow subteams to work at their own appropriate level of formality and with their own appropriate practices within the project. The defined mission of the project implies the intended use of products over which the project has responsibility. A project mission may be exploratory, for example to develop knowledge or skills, and is not intended to produce a deliverable product. The mission may be to support a pre-existing product which is delivered to customers, for example a legacy code. A project's mission may cover the full lifecycle of a product from inception through delivery. A single project may have multiple missions. For example, a software product may contain mature features (support mission), features under development (development mission) and research features that are not yet intended for customers (research mission). The organizational context of the project defines the functional roles and responsibilities, management responsibilities and authority, users and customers, and interrelated projects. Management includes the AQMC, Program Element Leads, and other line management as appropriate. An organization, a cooperating group of projects, or a framework project may share documention for shared practices, processes, and tools.
Practices:
PR1.   Document and maintain a strategic plan.   The mission (or scope) of the project is clearly defined, documented, and updated when the mission changes. Management responsibilities and authorities for the project are clearly defined, documented, and updated. The initial identification of project stakeholders and customers may also be addressed in the strategic plan. Commitments for changes to mission and organizational context are only negotiated by authorized personnel with appropriate technical inputs. This practice includes establishing authorities and beginning to identify sources of technical inputs.
Artifacts:
AR1   Strategic plan: [project's mission, management, stakeholders] (PR1)
Example Inputs:
•   Organization representatives (PR1)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Percentage of projects within an organizational unit with defined mission and management (PR1) ✓   Number of changes in the strategic plan related to mission, management, or stakeholders over a given period of time. (PR1)

4.2.2 Determination of Applicable Practices and Level of Formality

PROJECT MANAGEMENT **Determination of Applicable Practices and Level of Formality**
Overview Description:
Each ASC software project applies a risk-based assessment to determine its level of formality (LOF) in implementing applicable practices. This risk-based assessment considers factors such as project mission; customers' intended use; product or project complexity; project budget and schedule; stability of requirements and stakeholder organizations; and team dynamics and communications; and is performed with appropriate management and other stakeholder involvement. The appropriate ASC Program Element lead reviews and approves applicable practices and the level of formality. This assessment along with its review and approval is performed at the beginning of a project and the assessment is periodically reviewed to either verify that the conditions of the assessment have not significantly changed or to perform a reassessment when factors have changed significantly. A reassessment should initiate corrective actions to bring the project's LOF and applicable practices into compliance. The only risks addressed in this practice are those that can be mitigated by LOF and selection of software engineering practices. See section 4.2.5 for additional risk management practices.
Practices:
PR2. Perform a risk-based assessment, determine level of formality and applicable practices, and obtain approvals. ***Perform a risk-based assessment based upon the project's consequence of failure and likelihood of failure.*** Table 3 provides guidance in determining 'consequence of failure' once a project team has established its mission and obtained stakeholders' inputs. If different stakeholder inputs result in different consequence levels, the higher level should be given greater consideration. If the mission or stakeholder's inputs change significantly, a reassessment is warranted. The project team also uses Table 3 to guide its determination of its likelihood of failure to satisfactorily meet overall project commitments. The project estimates this likelihood by considering multiple factors relating to the software development team, the software product, and the environment. Likelihood of failure does not imply a mathematical probability. ***Decide on the applicable practices and level of formality that will mitigate the risk level.*** The project team uses the mission defined in its strategic plan (practice **PR1**) to guide its determination of which practices to implement. Depending upon its identified consequence level and its associated likelihood of failure to meet its commitments, each ASC software project may then tailor its implementation of the practices described in this Software Quality Plan as suggested by the intersection of its consequence of failure and its likelihood of failure. This tailoring will include a decision both on which practices are applicable and on the LOF to be applied in implementing these selected practices. An ASC program element lead reviews, approves, and tracks the project's assessed LOF and applicable practices for ASC projects in their domain. Table 4 presents the AQMC's expectations concerning applicable practices and appropriate level of implementation detail given a project's determined level of formality. A project may request a waiver from the AQMC's expectations. Such a waiver requires written approval from both the program element lead and the customer lead. Table 5 provides 'rules of thumb' on LOF issues related to artifacts, reviews, training, and tool usage.
Artifacts:
AR2 Approved level of formality and applicable practices (PR2)
Example Inputs:
• Customer and organization process requirements (PR2) **AR1** Strategic plan: [project's mission, management, stakeholders] (PR2)
Example Metrics/Measurements: AR4 Process and product metrics
✓ Percentage of LOF determinations that have been approved by the appropriate program element lead vs. total number of LOF determinations (PR2) ✓ Percentage of reevaluated and approved LOF determinations resulting from significant requirements changes vs. total number of significant requirements changes (PR2)

Table 3. Risk-Based Assessment to Determine Level of Formality.

Consequence of Failure		
Critical Potential for loss of human life, grave environmental damage, grave harm to the national or SNL's interest. Examples: • Weapon qualification decision, no test alternatives • Potential for Significant Finding Investigation	**High Level of Formality**	
High Potential for serious injury, serious environmental damage, serious harm to the national or SNL's interest. Examples: • Weapon qualification support, supplements tests • Potential for Significant Finding Investigation • Safety related	**Medium Level of Formality**	
Medium Potential for minor injury, minor environmental damage, minor harm to the national or SNL's interest, failure to make major milestones, or customer must go to great lengths to accommodate budget impact. Examples: • Early parts of Life Extension Projects provide a basis for refining design decisions • Design trade-off study		
Low No potential for injury, no environmental damage, no harm to the national or SNL's interest, failure to make minor milestones or minor budget impact. Examples: • Exploratory scoping (what is happening with this problem?)	**Low Level of Formality**	
	• Small and simple project • Requirements well-known & stable • Small team and good communication • Organization is stable	• Large and complex project • Requirements ill-defined or unstable • Large team and complex communication • Organization is unstable
	Likelihood of Failure	

AQMC Implementation Expectations Based Upon Determined Level of Formality

AQMC expectations on which practices should be implemented and at what level of implementation detail are presented in Table 4. ASC program element leads will likely furnish projects that fall under their domain further guidance for determining the appropriate level of formality at which they expect their projects to operate. A program element lead may decide to direct all projects in his/her purview to operate at a high level of formality; in which case, each project team would follow the expectations established in the (1) **High Level of Formality** column of the table. In some cases, the program element lead may request an individual project follow the steps outlined in practice **PR2** to determine its level of formality and then use this table to determine which practices it needs to follow and at what level of implementation detail. As explained in **PR2**, an approved waiver signed by the project's program element lead and customer element lead, if applicable, must accompany any exceptions to the expectations provided in Table 4.

Table 4. AQMC Implementation Expectations Based upon Determined Level of Formality.

Practice	Implementation Detail Symbols ● = high level of detail ◐ = medium level of detail ○ = low level of detail		
	(1)**High Level of Formality**	(2)**Medium Level of Formality**	(3)**Low Level of Formality**
Project Management (12)			
1. Strategic Planning			
PR1. Document and maintain a strategic plan.	●	◐	◐
2. Determination of Applicable Practices and Level of Formality			
PR2. Perform a risk-based assessment, determine level of formality and applicable practices, and obtain approvals.	●	●	●
3. Process Implementation and Improvement			
PR3. Document lifecycle processes and their interdependencies, and obtain approvals.	●	◐	○
PR4. Define, collect, and monitor appropriate process metrics.	●	◐	○
PR5. Periodically evaluate quality problems and implement process improvements.	●	◐	○
4. Requirements Engineering			
PR6. Identify stakeholders and other requirements sources.	●	●	◐
PR7. Gather and manage stakeholders' expectations and requirements.	●	◐	○
PR8. Derive, negotiate, manage, and trace requirements.	●	◐	○
5. Risk Management			
PR9. Identify and analyze risk events.	●	◐	○
PR10. Define, monitor, and implement the risk response.	●	◐	○
6. Project Planning, Tracking, and Oversight			
PR11. Create and manage the project plan.	◐	◐	○
PR12. Track project performance versus project plan and implement needed (corrective) actions.	◐	◐	○

Practice	Implementation Detail Symbols ● = high level of detail ◖ = medium level of detail ○ = low level of detail		
	(1)High Level of Formality	(2)Medium Level of Formality	(3)Low Level of Formality
Software Engineering (13)			
7. Software Development			
PR13. Communicate and review design.	●	◖	○
PR14. Create required software and product documentation.	●	◖	○
8. Integration of Third Party or Other Software			
PR15. Identify and track third party software products and follow applicable agreements.	●	◖	◖
PR16. Identify, accept ownership, and manage assimilation of other software products.	●	◖	○
9. Configuration Management			
PR17. Perform version control of identified software product artifacts.	●	●	◖
PR18. Record and track issues associated with the software product.	●	◖	○
PR19. Ensure backup and disaster recovery of software product artifacts.	●	●	◖
10. Release and Distribution Management			
PR20. Plan and generate the release package.	●	◖	○
PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution.	●	◖	◖
PR22. Distribute release to customers.	●	◖	○
11. Customer Support			
PR23. Define and implement a customer support plan.	●	◖	○
PR24. Implement the training identified in the customer support plan.	●	◖	○
PR25. Evaluate customer feedback to determine customer satisfaction.	●	◖	○
Software Verification (3)			
12. Software Verification			
PR26. Develop and maintain a software verification plan.	●	◖	○
PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.	●	◖	○
PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.	●	◖	○
Training (2)			
13. Training			
PR29. Determine project team training needed to fulfill assigned roles and responsibilities.	◖	◖	○
PR30. Track training undertaken by project team.	◖	◖	○

Table 5. Rules of Thumb for Level of Formality.

	Low Formality	**Medium Formality**	**High Formality**
Artifacts	Artifacts contain key details and may take the form of notes in an engineering notebook, hardcopy of drawing notes on a whiteboard, meeting notes, presentation materials, and email. Artifacts are available to at least the artifact owner and PI.	Artifacts contain significant detail, including key concepts and are likely in draft form. Artifacts are identified in the project plan and are stored in a repository available to all project team members.	Artifacts are complete and in final form. Artifacts are identified in the project plan. Format of the artifacts may include formal project, product, or process reports, or memos stored in a repository available to all project stakeholders.
Reviews	Takes the form of meeting notes, emails, and paired programming practices. Reviews are witnessed/approved as needed. Reviews consist of at least one reviewer who is knowledgeable and independent of artifact construction. Review records become artifacts.	Low formality plus PI and appropriate management are involved in reviews. Customers are informed of status of reviews. Key concepts of artifacts are reviewed and approved by team members and appropriate management. Review records become artifacts.	Reviews are scheduled in the project plan. Attendees may include management, PI, project team, subject matter experts and/or key stakeholders. Review results require approvals by appropriate management and stakeholders. Findings and issues are maintained in a formal report or issue tracking system. Review records become artifacts.
Training	Takes the form of mentoring and self-paced training, including reading books, journals, seminars, and self-study training material. Training records may include e-mail acknowledgement to team lead or PI. Team maintains a record of skills and training required to develop the skill set. Training records become artifacts.	Low formality plus identification of critical skills redundancy (where cross-training results in several team members who are knowledgeable of key areas). Feedback on effectiveness of training experiences is collected. Training records become artifacts.	Medium formality plus gathering of metrics for gauging effectiveness of training are identified, collected and applied. Training format may be extended to university and college degree programs, professional certifications, on and off-site classroom training, and computer-based training. Training records become artifacts.
Tools	Generic tools such as manual notebooks, calculators or common desktop tools such as office automation (word processing, spreadsheet, presentation, e-mail, project management). Key project members have access to these tools.	Low formality plus tools of a more specialized nature to address specific tasks (for example, DOORS for requirements management, SourceForge for collaborative environments). Tools are available to appropriate project members and appropriate management and stakeholders.	Medium formality plus all appropriate management and stakeholders have access. Ideally, selected tools are a program or corporate resource.

4.2.3 Process Implementation and Improvement

<table>
<tr><td colspan="1" align="center">PROJECT MANAGEMENT
Process Implementation and Improvement</td></tr>
</table>

Overview Description:

Process implementation typically includes the activities required to plan, define, implement, monitor, measure, and improve all aspects of a product lifecycle from concept to retirement. Examples of lifecycles include waterfall, iterative or spiral, and concurrent. Various methodologies can be employed to support software lifecycles. Practices are implemented through lifecycle processes which define the activities, interfaces, roles, and responsibilities. (See the Glossary for a definition of process.)

Process improvement is the continual activity to increase the ability of a process to meet its objectives. Lifecycle processes are evaluated by monitoring, measuring, and analyzing their effectiveness and efficiency with respect to their objectives. This evaluation is used to investigate alternative improvement solutions and select cost-effective improvements to the processes. An objective for process evaluation and improvement is to anticipate and prevent errors and nonconformance. Problems, errors, or nonconformance are analyzed to determine if corrective actions are required to improve the processes and prevent recurrence of similar problems. Process improvement changes are reviewed, managed, and documented.

These process implementation and improvement practices are treated separately from project planning, tracking, and oversight practices (section 4.2.5) to allow organizations to define common lifecycle processes that will be shared and followed by multiple projects. Otherwise, a project team may combine the implementation of these two practice areas.

For suggested effective metric and non-metric based process improvement techniques see Appendix F.

Practices:

PR3. Document lifecycle processes and their interdependencies, and obtain approvals.
 The project team defines and documents its applicable lifecycle processes by taking into consideration the level of formality, intended use, project objectives, cost, resource constraints, and compatibility with customers and other projects' activities. Defined lifecycle processes may include activities, interfaces, roles, and responsibilities. The appropriate stakeholders review and the appropriate management approves the documented lifecycle processes.

PR4. Define, collect, and monitor appropriate process metrics.
 The project team defines metrics to aid in the evaluation of process effectiveness and efficiency. Typically a new team identifies only selected metrics that will add immediate value in improving their processes or the way they approach their lifecycle activities. As the project evolves the number of metrics collected typically increases to address additional areas where improvements are needed.

PR5. Periodically evaluate quality problems and implement process improvements.
 Ideally the project team monitors conditions in order to investigate and prioritize alternative quality problem solutions. The team is responsible for documenting and implementing improvement solutions. Typically the project team analyzes metrics to aid in this evaluation.

Artifacts:

AR3 Approved project processes (PR3)
AR4 Process and product metrics (PR4)
AR5 Project process improvement actions (PR5)

Example Inputs:

- Customer and organization process requirements (PR3)
- Information on available and planned resources (PR3)

AR1 Strategic plan: [project's mission, management, stakeholders] (PR3)
AR2 Approved level of formality and applicable practices (PR3)

Example Metrics/Measurements: AR4 Process and product metrics

✓ Percentage of processes that are documented vs. processes identified to be documented (PR3)
✓ Also see also Software Verification; Project Planning, Tracking and Oversight; and Training

4.2.4 Requirements Engineering

PROJECT MANAGEMENT Requirements Engineering
Overview Description:
The purpose of requirements engineering practices is to capture, develop, validate, track, and control the product requirements. Product requirements typically span hardware, software, operations, support, documentation, product training, and other aspects. Requirements are based upon project mission, stakeholders' stated and implied needs, and organizational commitments. Although needs are not requirements they are considered along with requirements in order to improve quality. Changes to requirements must be managed throughout the lifetime of the project. Requirements are inputs to other practice areas. Risk management activities analyze and try to control events that affect the ability to satisfy requirements. Project planning determines whether and when requirements will be implemented. A product release identifies requirements that are newly satisfied in that release. Software verification reviews evaluate whether the product has met the requirements according to specified acceptance criteria. Requirements should be reviewed and approved by appropriate stakeholders.
Practices:
PR6. Identify stakeholders and other requirements sources. Sources of requirements potentially include stakeholders as well as regulatory, historical, organizational, and computational commitments. The project team communicates with the customers and other stakeholders regarding areas needing support. Stakeholders may also include suppliers of products that are to be integrated with the project product.
PR7. Gather and manage stakeholders' expectations and requirements. Product expectations and requirements are gathered from identified stakeholders, additional commitments, and submitted issues. The gathering activity may include identifying the source, criticality, priority, and acceptance criteria of the needs. There may be needs that are not clear. In these cases the originator should be contacted for further clarification. These sources may start out as a stockpile driver, expectations of fitness for intended use, a programmatic requirement, a physical or functional requirement, a modeling or simulation requirement, or an issue submitted against a previous version of derived software requirements.
PR8. Derive, negotiate, manage, and trace requirements. The software project team derives and negotiates software requirements based upon the gathered needs, and analysis of technical feasibility and resource availability. Negotiation optimally includes project team and stakeholder approvals of derived requirements and subsequent delivery commitments. The requirements are traced to product components that satisfy (forward tracing) or to verify that the requirement has been met (backward tracing). Changes to derived requirements and their associated status are managed and tracked. Ideally, requirements traceability supports analyzing the impact of the change.
Artifacts:
AR6 Product expectations and requirements (PR6, PR8) **AR7** Software requirements and attributes (PR7, PR8) **AR8** List of stakeholders and organizational commitments (PR6, PR8)
Example Inputs:
•   Stakeholder expectations and requirements (PR7) •   Organizational requirements (PR7) •   Platform requirements and characteristics (PR7) AR1   Strategic plan: [project's mission, management, stakeholders] (PR7) AR16 Managed issues: [enhancements, defects, questions, inquiries] (PR7, PR8) AR19 Customer support plan including training (PR7)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Cost of collecting, deriving, and managing requirements (PR6, PR7, PR8) ✓   Percentage of requirements added/changed in a time period vs. total number of requirements (requirements stability) (PR7, PR8) ✓   Percentage of requirements implemented in a time period with respect to number of requirements planned in a time period (PR8)

4.2.5 Risk Management

PROJECT MANAGEMENT Risk Management
Overview Description:
Risk management is the activity of identifying, addressing, and mitigating sources of risk before they become threats to successful completion of a project. A risk is a combination of the consequence and likelihood of an event. Risk management spans the lifetime of the project. The number of risks and risk factors is unbounded. Therefore, this practice area seeks to identify only primary and reasonably likely risks in the following areas: organizational, regulatory, technical, and project management. Risk management is intended to mitigate consequences and/or likelihood of these identified risk events. Monitoring risk events may be done in conjunction with the Project Planning, Tracking, and Oversight practices (see section 4.2.6).
Practices:
PR9. Identify and analyze risk events.   Significant risk events must be identified and clearly described before they can be analyzed and managed. As conditions change, identified risks should be reviewed and updated in a risk plan. An ideal risk analysis process identifies key attributes of each risk event such as the impact, likelihood, group(s) impacted by the risk event, and the organization (risk owner) responsible for any action associated with the risk event. Typically risk events are prioritized based on impact, likelihood, and potentially other factors.
PR10. Define, monitor, and implement the risk response.   A risk response is typically comprised of the risk disposition and corrective action(s) for events to be mitigated. Given a prioritized set of risk events, the project then determines the risk disposition of the highest priority events. Possible dispositions include *mitigate*, *transfer*, *accept*, and *avoid*. Teams may plan a response for unanticipated events that threaten the successful completion of the project. Projects monitor risk by collecting relevant information. The monitoring approach is documented in a risk plan and includes who does monitoring, how often, how information is collected, tools to assist monitoring, etc. If a risk event occurs, the planned corrective actions are implemented including notification of impacted stakeholders.
Artifacts:
AR9 Project plan [risks events, risk plan] (PR9) **AR10** Project reviews and needed (corrective) actions: [risk responses] (PR10)
Example Inputs:
• List of subject matter experts knowledgeable about potential risk events (PR9) AR6 Product expectations and requirements (PR9) AR7 Software requirements and attributes (PR9) AR8 List of stakeholders and organizational commitments(PR9) AR9 Project plan (PR9, PR10) AR10 Project reviews and needed (corrective) actions [tracking and oversight responses] (PR9) AR24 Technical reviews (PR9)
Example Metrics/Measurements: AR4 Process and product metrics
✓ Total number of identified risk events (provides some indication of the complexity of the software development project) (PR9) ✓ Percentage of prioritized risk events that can be mitigated with defined corrective actions vs. total number of prioritized risk events that can be mitigated (PR10) ✓ Number of risk events that were not anticipated but occurred (effectiveness of risk management planning) (PR9, PR10) ✓ Cost of implemented corrective actions during the monitoring cycle (PR10)

4.2.6 Project Planning, Tracking and Oversight

PROJECT MANAGEMENT Project Planning, Tracking, and Oversight
Overview Description:
The purpose of project planning, tracking, and oversight is to guide project implementation while balancing, monitoring, and analyzing project quality, cost (including cost of quality), schedule, and performance. Project planning includes preparing a plan that describes how the project will be performed and managed. The plan typically includes at least a statement of work, project constraints and goals, project deliverables, a project timeline, an assessment of required resources, and the availability of the resources. Many aspects of the project plan may already be captured by the ASC funding process. Various stakeholder organizations also use the project plan to fund, plan, and provide a basis for tracking and oversight. Updates to the project plan occur throughout the lifetime of the project. Tracking and oversight includes taking corrective actions as necessary. Corrective actions bring projected accomplishments and results back into compliance. Corrective actions could include adding resources to meet schedules, modifying the schedule, adding project budget, modifying cost criteria, and re-negotiating requirements or acceptance criteria.
Practices:
PR11. Create and manage the project plan. Project plans typically contain a project overview, project tasks, resource information, planning assumptions and constraints, dependencies, budget, schedule, and roles and responsibilities. This practice may include identifying tasks and evaluating feasibility, cost, resource requirements, and both internal and external dependencies of the tasks. See Appendix F for suggested tools to assist in project planning activities.
PR12. Track project performance versus project plan and implement needed (corrective) actions. The project team determines what project metrics are of interest, then monitors and analyzes these metrics. This monitoring may be performed via automated tools or manually and should take place frequently enough to allow time to analyze any significant variances prior to significant project impact. Once significant variances are identified, they are analyzed to determine their significance. For significant variances the root cause and potential corrective actions are determined. This activity may require discussion with stakeholders and management concerning the severity and impact of the identified variances.
Artifacts:
AR9   Project plan: [risk events, risk plan, overview, milestones, task list, resource information, roles and responsibility assignments, assumptions, constraints, dependencies, budget, schedule, SCM plan, etc.] (PR11) AR10 Project reviews and needed (corrective) actions: [risk responses, tracking and oversight responses] (PR12)
Example Inputs:
•   Information on available resources (PR11) AR1   Strategic plan: [project's mission, management, stakeholders] (PR11) AR3   Approved project processes (PR12) AR4   Process and product metrics (PR12) AR6   Product expectations and requirements (PR11) AR7   Software requirements and attributes (PR11) AR8   List of stakeholders and organizational commitments (PR12) AR9   Project plan: [risk plan, risk events](PR12) AR19 Customer support plan including training (PR11)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Percentage of tasks directly tied to one or more requirement vs. total number of tasks (PR11) ✓   Percentage of actual vs. planned budget, schedule, performance (PR12) ✓   Number of corrective actions taken in a specified time frame (PR12)

4.3   Software Engineering

Software engineering is a systematic approach to the specification, design, development, test, operation, support, and retirement of software. The software engineering activities identified in this section are Software Development, Configuration Management, Release and Distribution Management, and Customer Support.

4.3.1 Software Development

SOFTWARE ENGINEERING Software Development
Overview Description:
The purpose of the software development processes is to generate a correctly working product for the customer; this product is often, but not always, software. Generally, software development processes include design, implementation, and testing of the software products or reuse of existing implementations. Other practices related to software development are covered elsewhere: Requirements Engineering activities in section 4.2.4, Configuration Management activities including version control and issue tracking in section 4.3.3, and Software Verification activities for reviews and testing in section 4.4. The lifecycle processes are documented in section 4.2.3 Process Implementation and Improvement. The Software Quality Plan prescribes no specific lifecycle or any particular software development methodology.
Practices:
PR13. Communicate and review design.   Design is the process of defining architecture, components, interfaces, and other characteristics of a system or components. Design activities transform requirements into artifacts that are used for the development of software. Design artifacts capture information and process specifications that document dependencies, information flows, algorithms, the interfaces, and all the components. These help ensure requirements are implemented and team members have a common understanding of the design. The impact of implementation choices on design is continuously incorporated. Relevant stakeholders are informed of issues and included in decisions. Documentation of a design supports development, product maintenance, tracing of requirements, verification, and end users. Design reviews are an important aspect of software development. Depending upon the software methodology being used by a project team, design artifacts may not be simultaneously available for formal reviews so informal design reviews and design artifacts may provide the quality necessary for this practice. See Table 5 for suggestions on carrying out level of formality for artifacts and reviews.
PR14. Create required software and product documentation.   The project team creates the required product artifacts (such as code, user documentation, developer's guide, and installation guide) using the documented project processes. Note that testing of these products is part of software verification. These artifacts implement the requirements and are updated to reflect the "as built" product.
Artifacts:
AR11 Design artifacts: [documentation and/or reviews] (PR13) **AR12** Implementation artifacts: [software code, assimilated other software, design documents, user documentation, developer's guide, installation guide, theory manual, interface manual, etc.] (PR14)
Example Inputs:
• External knowledge (subject matter experts, algorithms, technical reports) (PR13, PR14) • Assimilated software (from a source outside the project) (PR14) AR3   Approved project processes (PR13, PR14) AR6   Product expectations and requirements (PR13, PR14) AR7   Software requirements and attributes (PR13, PR14) AR8   List of stakeholders and organizational commitments (PR13) AR9   Project plan (PR13, PR14) AR13 Identification and acquisition records (PR13, PR14) AR16 Managed issues: [enhancements, defects, questions, inquiries] (PR13, PR14) AR17 Release specification (PR14)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Percentage of requirements implemented versus total number of requirements planned for release (PR13, PR14) ✓   Percent of test cases successfully executed versus all test cases (PR27) ✓   Number of defects resolved versus all defects discovered (PR27)

4.3.2 Integration of Third Party or Other Software

SOFTWARE ENGINEERING Integration of Third Party or Other Software
Overview Description:
Projects use or incorporate third party or other existing software products in order to satisfy needed capabilities without incurring the cost of redeveloping those capabilities. Such software may be a simple library, an integrated set of libraries, compilers and linkers, or even an operating system. Sources of such software may be commercial, open source, other ASC or SNL projects, or research efforts. This practice area focuses on integration activities such as identifying, tracking, establishing trust in, assimilating, or honoring agreements (for example, protecting intellectual propery) for third party or other existing software products. Note that requirements traceability (practice **PR8**) should include tracing requirements satisfied through the integrated third party or other existing software.
Practices:
PR15. Identify and track third party software products and follow applicable agreements.   A project typically uses third party software products without modification. However, if the project does modify the third party software those modifications must be tracked until the supplier incorporates those modifications into the third party software. A project may acquire and configuration manage software (for example, public domain software) or may use software as-is in the computational environment (for example, a compiler). A third party software product, its source, and the project's basis for trust in that product should be identified. A basis for trust could be simply noting the supplier's long-standing reputation or confirming that another trusted project has already established trust in the third party software, or could involve more complex verification efforts. Applicable agreements with a third party software product supplier could include licenses, protection of intellectual property, or customer support.
PR16. Identify, accept ownership, and manage assimilation of other software products.   Existing software may be assimilated into a project such that the project team accepts responsibility for maintaining, supporting, and potentially continuing development of the software. Assimilation should consider the effort needed to ensure that the software meets the project's verification and other software quality practices and standards. Assimilation should also consider the potential impact to the project's mission, applicable practices, and level of formality.
Artifacts:
AR12 Implementation artifacts: [assimilated other software] (PR16) **AR13** Identification and acquisition records (PR15, PR16)
Example Inputs:
•   Third party software (PR15) •   Other software (PR16) AR22 Software verification plan (PR15, PR16)
Example Metrics and Measurements: AR4 Process and product metrics
✓   Time and effort spent integrating other software products into environment (PR15, PR16)

4.3.3 Configuration Management

SOFTWARE ENGINEERING Configuration Management
Overview Description:
The purpose of configuration management (CM) is to provide a controlled environment for development, production, and support activities. CM includes identifying which software product artifacts are to be managed; maintaining version controlled baselines of these artifacts; providing an issue tracking system for recording associated issues or change requests related to product artifacts; and tracking the status of these issues throughout the project's lifetime. Configuration management must ensure retrieval of any baselined artifact over the project's lifetime. Note: some specific artifacts (records) and their retention schedule may be subject to SNL's Record Management Policies.
Practices:
PR17. Perform version control of identified software product artifacts.   As part of version control project teams typically identify project artifacts that will be kept in a repository, access and version control those artifacts, create and recover product baselines, and manage changes to these baselines.
PR18. Record and track issues associated with the software product.   This practice typically includes a process (change management) of recording and tracking all appropriate changes that occur to identified software product artifacts, including requirements, throughout their lifetime. Issue tracking typically includes an issue classification scheme and allows for the submittal of enhancement requests, problem and defect reports, and inquiries. Customers are a source of submitted issues. Section 4.3.5 Customer Support addresses customer issue submission and response.
PR19. Ensure backup and disaster recovery of software product artifacts.   This practice ensures backup is performed and disaster recovery of software product artifacts and associated baselines is possible should the repository become unavailable or destroyed. Backup and recovery capability includes the identification of where product artifacts are stored, a defined schedule for when backups are made, and a method of recovering or restoring backups should a disaster occur. The disaster recovery capability should be periodically tested to ensure that artifacts can be recovered and restored with minimal disruption to other project activities. This practice may be satisfied through confirmation that system administration is performing backups, ensuring safe storage, and testing recovery.
Artifacts:
AR14 Version controlled records, including baselines and associated configurations (PR17) **AR15** Backup records and recovery test results (PR19) **AR16** Managed issues: [product quality results (for example, non-conformances), enhancements, defects, questions, inquiries] (PR18)
Example Inputs:
•   Customer issues (PR18) AR1-AR26 Appropriate product artifacts (PR17, PR18)
Example Metrics and Measurements: AR4 Process and product metrics
✓   Percentage of software product artifacts stored under version control vs. total number of software product artifacts identified for versioning (PR17) ✓   Success rate of disaster recovery vs. total disaster recoveries attempted (PR19) ✓   Number of issues closed, deferred, or left open compared to total number submitted in a given period of time (PR18)

4.3.4 Release and Distribution Management

SOFTWARE ENGINEERING Release and Distribution Management
Overview Description:
The purpose of the release and distribution practices is to manage versions of the software product that are distributed to customers. Release management includes handling the requests for a release as well as preparation of the release. A release may include all elements of the product or a defined subset of the product. When the project team has completed all artifacts necessary for a release the team creates a baseline in preparation for distribution. The baselined product undergoes release certification before being distributed and supported. Release certification ensures that all release criteria are satisfied, that identified release artifacts are adequately reviewed, and that all planned testing is completed and satisfactory.
Practices:
PR20. Plan and generate the release package. 　　This practice includes determination of the release criteria such as: the release contents, dependencies on external products, targeted distribution date, required resources, and internal activities for completion of the release. Release contents may include code, user guides, training material, theory manuals, installation notes, and test cases that the customer can run to check installation. Internal activities may include reviews, installation testing, and generation of release notes. Release notes may include a running history of other releases associated with the project.
PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution. 　　Release certification ensures that all release criteria are satisfied. Certification may be a multi-step process to ensure that the release has been sufficiently verified to be distributed. A final review should verify that all required artifacts exist and are associated with the correct version number.
PR22. Distribute release to customers. 　　In distributing the release to customers the project team may consider whether any license agreements need to be updated, whether the product falls under export control restriction, and whether certain types of customers (for example, those providing funding) need special instructions or support. The project team may also decide to notify appropriate customers that a previous version of the product is being retired.
Artifacts:
AR17 Release specification (PR20) **AR18** Product release package (bill of materials, release notes, certification, software, etc.) (PR20, PR21, PR22)
Example Inputs:
• 　　Internal/external request for a release (PR20) • 　　Identified customers for whom release is intended (PR22) • 　　List of target platforms for the release (PR22) • 　　Information for release notes (PR20, PR22) • 　　Product artifacts that will be included in the release (PR20, PR22) AR23　Test artifacts: [test cases, test results] (PR21) AR24　Technical reviews (PR21)
Example Metrics/Measurements: AR4 Process and product metrics
✓ 　　Percentage of releases completed by their planned release date vs. number of releases planned (PR20) ✓ 　　Time and effort required to certify a release on a particular platform (PR21) ✓ 　　Number of issues, by severity, reported with each identified release (PR22)

4.3.5 Customer Support

SOFTWARE ENGINEERING Customer Support
Overview Description:
The purpose of this area is to assist and train customers in the installation, operation, and ongoing use of the product. Customer support also includes those activities required to manage feedback concerning the product. Each project team defines and implements a customer support plan to address the needs and expectations of appropriate customers, for example, those customers with organizational commitments. The customer support plan may be a single shared agreement intended to address the needs of multiple customers. Resources for implementing this plan should factor into project planning.
Practices:
PR23. Define and implement a customer support plan. A customer support plan may specify the period of support, responsibilities, point of contact for questions on any aspect of the product release, commitment to deliver documentation and training, and other support deemed necessary. A support plan typically includes a feedback process for the submittal, prioritization, and timely resolution of issues associated with the product. The feedback process may utilize the issue tracking process defined in configuration management. The customer support plan may also include information related to product installation, supported platforms, consistent product interfaces, and frequency of product installations. Customers who intend to provide funding for support activities are likely to be included in negotiations as to what will be included in the plan.
PR24. Implement the training identified in the customer support plan. During requirements gathering the project team typically determines the details of the product training plan that includes requisite documentation. Training may be developed as a formal class or self-study material. Topics covered by training may include installation, use, theory manuals, tutorials, and tests. Ideally project teams maintain records (such as class, attendees, and dates) for training they deliver.
PR25. Evaluate customer feedback to determine customer satisfaction. The ultimate measure of quality is customer satisfaction. At the appropriate point in the product's lifecycle, the project team may decide to solicit customer feedback regarding the level of satisfaction with the product and the support the team provides. This information is used to support identification of systemic quality problems and opportunities for process improvement.
Artifacts:
AR16 Managed issues (PR23, PR24, PR25)
AR19 Customer support plan including training (PR23)
AR20 Customer training records (PR24)
AR21 Customer satisfaction evaluation (PR25)
Example Inputs:
AR6   Product expectations and requirements (PR23, PR24)
AR8   List of stakeholders and organizational commitments (PR23, PR24, PR25)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Average time spent resolving customer support issues (PR23, PR24)
✓   Degree of customer satisfaction with requirements that have been implemented (demonstrates effectiveness of the process for capturing expectations and requirements) (PR25)

4.4   Software Verification

Some ASC code teams have participated in the development of a verification and validation (V&V) plan and perhaps have performed some of the activities outlined in this plan. Information from an existing V&V plan can potentially be leveraged for the software verification practices. V&V plans include the test planning related to a verification test suite and technical reviews. If a code team has a test plan but no software verification plan, the test plan can be enhanced with planning information for technical reviews.

SOFTWARE VERIFICATION
Overview Description:
The purpose of software verification is to ensure (1) that specifications are adequate with respect to intended use and (2) that specifications are accurately, correctly, and completely implemented. Software verification also attempts to ensure product characteristics necessary for safe and proper use are addressed. Software verification occurs throughout the entire product lifecycle.   Software verification activities are an integral part of software development, operation, and support practices. In this context, the goal is to detect potential problems as early as possible. Software artifacts to be verified typically include specifications, requirements, design, code, third party libraries, software verification plan, test cases, product documentation, and training package. If these artifacts are changed, retesting and reevaluation of the changes will need to occur.   In addition to software verification, both QC-1 and ISO 9000 refer to "validation" activities. Generally, these standards define validation activities as helping to assure that "you built the right thing". Validating a complex software product (such as a modeling and simulation code) requires a broad set of tasks and participation from a number of different communities: experimental, analysis, code development, and customer. For some ASC program elements, this wide-ranging scope of activities is the responsibility of the Verification and Validation (V&V) program element. One of the project teams' contributions to validation activities includes software verification. Validation activities referenced in QC-1 and ISO 9000 include (a) evaluating whether the negotiated requirements, when implemented, adequately support the customer's mission and (b) testing to the negotiated requirements.
Practices:
PR26. Develop and maintain a software verification plan.   This practice typically involves identifying the list of artifacts to be reviewed, a list of knowledgeable reviewers, test and technical review approach, tools, associated verification test cases. Other information, which may appear instead in a project plan, includes schedules for tests and technical reviews, resources, and responsibilities. The software verification plan includes tests and reviews that demonstrate that requirements are being met and acceptance criteria that are used in the review of test results. Optimally the software verification plan addresses (1) the types of tests (see Appendix E); (2) when test results are reviewed; (3) the technical reviews to be performed and their objectives; and (4) the technical review schedule.
PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.   Testing occurs throughout the product lifecycle. Ideally, results from performing tests found in the software verification plans or in separate test cases may be reviewed with respect to each test's associated acceptance criteria. Test results form the basis for later reviews or concerns that may arise regarding verification of the software product. See Appendix E for a discussion of test terms and test categories.
PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.   These reviews may also include evaluation of adequacy with respect to intended use and acceptance criteria. Acceptance criteria could include comparison tests with analytic solutions or other pedigreed codes, traceability analysis to determine support of the requirements for each critical artifact,

SOFTWARE VERIFICATION
interface analysis to check consistency and completeness of the user interface, data flow such as unit conversion, and control flow between components represented by the artifact. Independent technical reviews include some participants that are independent of the creation of the item or activity being reviewed and knowledgeable in relevant subject areas.
Artifacts:
AR16   Managed issues [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries] (PR27, PR28) **AR22**   Software verification plan (PR26) **AR23**   Test artifacts [test cases, test results] (PR27) **AR24**   Technical reviews (evidence that review occurred and review results) (PR28)
Example Inputs:
AR1   Strategic plan: [project's mission, management, stakeholders] (PR26) AR6   Product expectations and requirements (PR26, PR27, PR28) AR7   Software requirements and attributes (PR26, PR27, PR28) AR8   List of stakeholders and organizational commitments (PR26, PR27, PR28) AR11   Design artifacts (PR27, PR28) AR12   Implementation artifacts: [including assimilated other software] (PR27, PR28)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Percentage of requirements tested vs. total number of requirements (PR27) ✓   Number of defects in the released product not caught by software verification activities prior to the release (PR26, PR27, PR28)

4.5   Training

Training spans all three SQE areas outlined in the GP&G (see Figure 2) and addresses the importance of the "human asset" in ASC software development.

TRAINING
Overview Description:
The goal of training is to enhance the skills and motivation of a staff that is already highly trained and educated in the areas of scientific software development, algorithms, and/or computer science. This practice addresses training needs of the project teams especially for, but not limited to, following the project teams' process implementation. The purpose of training is to develop the skills and knowledge of individuals and teams so they can fulfill their process and technical roles and responsibilities. Project teams need to ensure that the training needs of the project are satisfied in accordance with their project plan. Customer training is addressed in Customer Support section, 4.3.5.
Implementing these two training practices typically includes preparing a list of required and desired training to be taken, when the training is needed, the acceptable methods of receiving the training (for example, mentoring, classroom setting, online course, etc.), when the training is actually taken, and metrics for gauging the effectiveness of the training.
Practices:
PR29. Determine project team training needed to fulfill assigned roles and responsibilities.   Training needs may be determined by comparing the actual skills and knowledge of the team members to the skills and knowledge necessary to complete their roles and responsibilities. Training needs may also result from organizational training requirements.
PR30. Track training undertaken by project team.   Project team members undertake their planned training. The project team may maintain training records indicating training that the project team members participated in, when the training occurred, and the measurements and/or metrics associated with the training.
Artifacts:
AR25 Project team training needs (PR29) **AR26** Project team training records (PR30)
Example Inputs:
•   Organization training requirements and opportunities (PR29) AR3   Approved project processes (PR29) AR9   Project plan: [task list, resource information, roles and responsibility assignments] (PR29)
Example Metrics/Measurements: AR4 Process and product metrics
✓   Percentage of identified training needs satisfied versus total training needs (PR30) ✓   Cost of training (time, materials, and travel) (PR30) ✓   Objective ratings to measure training effectiveness (PR29, PR30)

4.6   Summary of Practices and Artifacts

Table 6 provides a list of the practices with the artifacts generated by those practices.

Table 6. Practices and Generated Artifacts.

Practice Number	Practice Description	
	Artifact Number	**Artifact Description**
PR1	**Document and maintain a strategic plan.**	
	AR1	*Strategic plan: [project's mission, management, stakeholders]*
PR2	**Perform a risk based assessment, determine level of formality and applicable practices, and obtain approvals.**	
	AR2	*Approved level of formality and applicable practices*
PR3	**Document lifecycle processes and their interdependences, and obtain approvals.**	
	AR3	*Approved project processes*
PR4	**Define, collect, and monitor appropriate process metrics.**	
	AR4	*Process and product metrics*
PR5	**Periodically evaluate quality problems and implement process improvements.**	
	AR5	*Project process improvement actions*
PR6	**Identify stakeholders and other requirements sources.**	
	AR6	*Product expectations and requirements*
	AR8	*List of stakeholders and organizational commitments*
PR7	**Gather and manage stakeholders' expectations and requirements.**	
	AR7	*Software requirements and attributes*
PR8	**Derive, negotiate, manage, and trace requirements.**	
	AR6	*Product expectations and requirements*
	AR7	*Software requirements and attributes*
	AR8	*List of stakeholders and organizational commitments*
PR9	**Identify and analyze risk events.**	
	AR9	*Project plan: [risk events, risk plan]*
PR10	**Define, monitor, and implement the risk response.**	
	AR10	*Project reviews and needed (corrective) actions: [risk responses]*
PR11	**Create and manage the project plan.**	
	AR9	*Project plan: [risk events, risk plan, overview, milestones, task list, resource information, roles and responsibility assignments, assumptions, constraints, dependencies, budget, schedule, SCM plan, etc.]*
PR12	**Track project performance versus project plan and implement needed (corrective) actions.**	
	AR10	*Project reviews and needed (corrective) actions: [risk responses, tracking and oversight responses]*
PR13	**Communicate and review design.**	
	AR11	*Design artifacts: [documentation and/or reviews]*
PR14	**Create required software and product documentation.**	
	AR12	*Implementation artifacts: [software code, assimilated other software, design documents, user documentation, developer's guide, installation guide, theory manual, interface manual etc.]*
PR15	**Identify and track third party software products and follow applicable agreements.**	
	AR13	*Identification and acquisition records*
PR16	**Identify, accept ownership, and manage assimilation of other software products.**	
	AR12	*Implementation artifacts: [assimilated other software.]*
	AR13	*Identification and acquisition records*
PR17	**Perform version control of identified software product artifacts.**	
	AR14	*Version controlled records, including baselines and associated configurations*

Practice Number	Practice Description	
	Artifact Number	**Artifact Description**
PR18	**Record and track issues associated with the software product.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
PR19	**Ensure backup and disaster recovery of software product artifacts.**	
	AR15	*Backup records and recovery test results*
PR20	**Plan and generate the release package.**	
	AR17	*Release specification*
	AR18	*Product release package (bill of materials, release notes, certification, software, etc.)*
PR21	**Certify that the software product (code and its related artifacts) is ready for release and distribution.**	
	AR18	*Product release package (bill of materials, release notes, certification, software, etc.)*
PR22	**Distribute release to customers.**	
	AR18	*Product release package (bill of materials, release notes, certification, software, etc.)*
PR23	**Define and implement a customer support plan.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
	AR19	*Customer support plan including training*
PR24	**Implement the training identified in the customer support plan.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
	AR20	*Customer training records*
PR25	**Evaluate customer feedback to determine customer satisfaction.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
	AR21	Customer satisfaction evaluation
PR26	**Develop and maintain software a verification plan.**	
	AR22	*Software verification plan*
PR27	**Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
	AR23	*Test artifacts: [test cases, test results]*
PR28	**Conduct independent technical reviews to evaluate adequacy with respect to requirements.**	
	AR16	*Managed issues: [product quality results (for example, non-conformance), enhancements, defects, questions, inquiries]*
	AR24	*Technical reviews (evidence that review occurred and review results)*
PR29	**Determine project team training needs to fulfill assigned roles and responsibilities.**	
	AR25	*Project team training needs*
PR30	**Track training undertaken by project team.**	
	AR26	*Project team training records.*

5   Assessment Strategy for Conformance to ASC Practices

Assessments of project teams' process implementation and of their compliance with the practices identified in this Software Quality Plan will be performed with the following strategy:

- ASC program element leads sponsor assessments and decide on the overall goals and objectives for each assessment.
- The assessment sponsor assembles an independent team to develop an appropriate approach and assessment tool to achieve the stated assessment goals and objectives. The independent assessment team must be technically qualified and knowledgeable according to education, training, and experience.
- Project teams perform a self-assessment which establishes implementation priorities for the individual teams.
- The assessment sponsor authorizes an independent team to perform assessments.
- Results of the self-assessment and independent assessment are published and presented to the assessment sponsor.
- The assessment sponsor communicates best practices identified from the assessments to the project teams.

For the purpose of this software quality plan, assessments fall into two categories: large-scale and small-scale. Large-scale assessments include independent program-level assessments conducted across all required elements and practices. Small-scale assessments evaluate a limited number of project teams and/or practices. The type, frequency, and scheduling of assessments is determined by ASC management. Assessment artifacts include the assessment procedure as well as an assessment report.

ASC management will direct project teams to periodically perform internal self-assessments to compare their current practice implementations to management defined goals and associated criteria. This approach will help the teams to determine those areas in which they are making good progress or, alternatively, in which they may need to focus improvement efforts. In addition to identifying areas that are appropriate for increased improvement efforts, the software project teams can observe how they are improving over time by comparing previous assessments to current assessments.

Project teams involved in independent assessments will want to focus on ensuring that documented processes for the various practices are accessible and being followed. The teams will also need to be able to furnish project artifacts that demonstrate that they are following their defined processes. In addition, team members involved in assessment interviews will be asked to explain how their project operates, whether processes are in place, and how consistently they are following these processes.

A checklist that can be used as an assessment tool is included in Appendix D. This checklist is applicable for both self and independent assessments practice implementation.

References

Required. The following are upper-tier documents that specify quality requirements for this site-specific practices document:

1. *Corporate Process Requirement No. CPR001.3.2, Corporate Quality Assurance Program*, Sandia National Laboratories, August 2003.

2. *Corporate Process Requirement No. CPR001.3.6, Corporate Software Quality Assurance*, Sandia National Laboratories, December 2001.

3. *Department of Energy, DOE/AL Quality Criteria (QC-1),* Revision 9, February 5, 1998. Available at http://prp.lanl.gov:8686/.

4. Hodges, A., G. Froelich, D. Peercy, M. Pilch, J. Meza, M. Peterson, J. LaGrange, L. Cox, K. Koch, N. Storch, C. Nitta, and E. Dube, Department of Energy, *ASCI Program Software Quality Engineering: Goals, Principles, and Guidelines,* DOE/DP/ASC-SQE-2000PFDRFT-VERS2, Albuquerque, NM, February 2001.

Guidance. The following are documents that provide additional information that is useful in developing and implementing Sandia ASC SQE policies and practices:

5. Berg, R. and A. Hodges, *ASCI Simulation and Computer Science (S&CS) and Ongoing Computing Software Quality Plan,* Version 2, March 2003.

6. *The Capability Maturity Model Guidelines for Improving the Software Process*, Software Engineering Institute, 1995.

7. *CMMI-SE/SW/IPP/SS v1.1*, Software Engineering Institute, March 2002.

8. Ellis, M., C. M. Williamson, J. Schofield, and L. Bonano, *2003 SNL ASCI Applications Software Engineering Assessment Report,* SAND2004-0075, Sandia National Laboratories, February 2004.

9. Hodges, A. L., G. K. Froehlich, M. Pilch, and D. E. Peercy, *Risk Management Plan – Sandia National Laboratories ASCI V&V Program*, SAND2002-1048, April 2002.

10. *IEEE Std.610.12-1990, IEEE Standard Glossary of Software Engineering Terminology,* Standards Coordinating Committee of the IEEE Computer Society, USA, December 2000.

11. *ISO 9000:2000 Quality Management Systems – Fundamentals and Vocabulary* and *ISO 9001:2000 Quality Management Systems – Requirements*, International Standards Organization, December 2000.

12. Pilch, M., T. Trucano, J. Moya, G. Froehlich, A. Hodges, and D. Peercy. *Guidelines for Sandia ASCI Verification and Validation Plans - Content and Format: Version 2.0,* SAND2000-3101. Albuquerque: Sandia National Laboratories, January 2001.

13. Program Management Institute (PMI) Standard: A Guide to the Project Management Book of Knowledge (PMBOK Guide) 2000 Edition, Project Management Institute, Newtown Square, Pennsylvania.

14. *Software Information Life Cycle*, Center 9500, http://www-irn.sandia.gov/silc, Sandia National Laboratories, September 2003.

15. Zepper, J., K. Aragon, M. Ellis, D. Eaton, and K. Byle, Kathleen, *ASCI Applications Software Quality Engineering Practices, Version 2,* SAND2003-0962, Sandia National Laboratories, April 2003.

Appendix A   Glossary and Acronyms

Glossary

acceptance criteria   The criteria that a system or component must satisfy in order to be accepted by a user, customer, or other authorized entity.

artifact   A documented process, deliverable or work product. A configuration-controlled artifact is stored in a corporate repository (library) and changes to it are controlled via reported issues.

assessment   An appraisal by a trained team of software professionals to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing that organization, and to obtain the organizational support for software process improvement.

baseline   A set of specifications or artifacts that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through change control procedures.

benchmarking   A quality tool used to periodically and continually measure and compare an organization's work processes with those in competing or similar organizations. The goal of benchmarking is to increase the organization's performance by adopting the best practices of industry leaders.

best practices   Those activities that have proven to be of high value, have improved quality, have improved productivity, or have enhanced customer satisfaction. Typically, these practices are measured activities or have metrics to show their value and are leveraged across an organization.

critical path method   A network analysis technique used to predict project duration by analyzing which sequence of activities (which path) has the least amount of scheduling flexibility.

customer   A collective term that may include the end user of the proposed system, the funding agency, the acceptor who will sign-off delivery, and the managers who will be responsible for overseeing the implementation, operation, and maintenance of the system.

customer support   The assistance, training, and documentation a project provides to ensure that the customer is satisfied and able to use the product as intended. Typically, a support plan is drawn up to specify what will, and what will not, be provided by the project team and for what period of time.

defect   (1) A flaw in a system or system component that causes the system or component to fail to perform its required function. (2) Non-fulfillment of a requirement related to an intended or specified use.

design of experiments   An investigation carried out in a planned manner and which relies on a statistical assessment of results to reach conclusions at a stated level of confidence. DOE is particularly useful for investigating complex systems whose outcome may be influenced by a potentially large number of factors.

error-proofing   Also known as fool-proofing, mistake-proofing and Poka-Yoke (Japanese quality term) An example of error-proofing for software development is a process checklist. The checklist prevents errors from missing an activity or performing the activity in the wrong sequence.

gantt   a graphic display of schedule-related information (sometimes called a bar chart).

interface analysis   The evaluation of presentation and flow (control and data) between components represented by the artifact.

issue   A point of concern, a problem, or a comment that is raised in regard to a practice of a software lifecycle area. The issue is a form of feedback and will usually be specific to an artifact suggesting rework, improvement, or enhancement.

level of formality   The degree of detail, form, and frequency to which a project defines and carries out its process for implementing a practice.

lifecycle   The period of time that begins when a software product is conceived and ends when the software is no longer available for use. Typically a lifecycle includes concept, requirements, design, implementation, test, installation, and operation and maintenance phases. These phases may overlap or be performed iteratively.

lifecycle model   An approach to the lifecycle that provides adequate detail of the order and phases. Some examples include spiral, evolutionary, sequential, and iterative.

measure   A unit of measurement (such as source lines of code or document pages of design).

measurement   The dimension, capacity, quantity, or amount of something (for example, 300 source lines of code or 7 document pages of design).

metric   A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

mitigate   Reduce the probability and/or impact of a risk to below an acceptable threshold.

policy   An accepted principle, established by decision makers, to direct and influence the activities of those to whom the policy pertains.

practice   A set of activities identified for accomplishing some portion of the required areas identified in the ASC Software Quality Plan.

process   A set of steps performed for a given purpose (for example, implementation of a practice). A well-documented process contains inputs, outputs, roles and responsibilities, sequences and dependencies, reviews and approvals, and entry and exit criteria. A process should have many but not necessarily all these attributes. It may be textual or graphical but should not be merely imaginary or virtual.

process metric   This type of metric measures the characteristics of the overall development process, such as the number of defects found throughout the process during different kinds of reviews.

product metric This type of metric is a measurement of an intermediate or final product of software development and, therefore, addresses the output of a software development activity. Examples of such metrics are a size metric for the number of requirements and a complexity metric for software.

production software   This type of software is implemented in a production environment, characterized as stable (meaning changes are recorded and analyzed), and fully supported by the project development team.

program evaluation and review technique   An event-oriented network analysis technique used to estimate program duration when there is uncertainty in the individual activity duration estimates.

quality   (1) Conformance to customer requirements and expectations. (2) The degree to which a system, component, or process meets specified requirements. (3) The degree to which a system, component, or process meets customer or user needs or expectations.

records management   SNL has a formal records management program that can be accessed at: http://www-irn.sandia.gov/recordsmgmt/rmm/rmmframe.html. SNL records are defined to "include any recorded information or documentation (including books, papers, maps, photographs, microfilm, or electronic media) created or received and used in the technical and administrative work. This website gives information on determining what is, and what isn't a 'record' as well as information on the responsibilities of employees to protect and manage such records." (Sandia Records Management)

regression test   Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

release   A snapshot in time of a software product available for distribution. Typically includes software as source or executable.

release plan   A plan prepared and followed by the project team specifying what needs to be accomplished for releasing the next version of a software product. The release plan typically specifies what the release will contain; what the release depends on externally such as compilers, version of required utility, etc.; when the release will be ready for distribution; what resources will be needed to

prepare the release; and other dependencies for completing the release (for example, installation testing, user documentation, reviews, training, and release notes).

requirement   A need or expectation that is stated, generally implied, or obligatory.

review   A quality assurance activity that establishes confidence in codes and supports software verification. Types of reviews are as follows:

- **management -** An evaluation performed to verify that *commitments* for the specified activities have been satisfied.
- **quality -** An evaluation performed to verify compliance with *process and artifact* requirements.
- **technical -** An evaluation to determine if the *content* of the item submitted for review conforms to the requirements.

reviewer   An independent person qualified to perform a review.

risk   A combination of the likelihood of an event's occurrence and its impact.

risk mitigation   Reduce the probability and/or impact of a risk to below an acceptable threshold and/or increasing the positive consequence.

risk plan   This document details all identified risks including description, cause, probability of occurring, impact(s) on objectives, proposed responses, owners, and current status. The plan also addresses procedures and techniques to enhance opportunities and to reduce threats to the projects' objectives.

role   A set of defined responsibilities that may be assumed by one or more individuals.

software engineering   The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

software product   Any software project deliverable given to the customer. One software product is typically the code (executable and/or source).

software quality assurance   (1) A set of activities designed to evaluate the process by which products are developed. (2) Planned and systematic actions necessary to provide adequate confidence that the software product conforms to established requirements.

software quality engineering   The practices a software team follows to ensure that quality standards are incorporated into its software product.

software verification   (1) Ensures that requirements are accurately, correctly, and completely (with respect to the delivery commitments) implemented throughout the entire product lifecycle, and that requirements are adequate from the intended uses of the software. (2) The process of determining whether or not the mathematical formulation is solved correctly, that is, whether the computer simulation correctly represents the conceptual model and its solution.

stakeholder   Individuals and organizations (internal and external) that are actively involved in a project or whose interests could impact or may be affected as a result of project execution or project completion. Customers, users, and project team members are stakeholders.

subject matter expert   An individual who is responsible for providing guidance and information to the software project team in areas or topics outside the scope of the team's expertise.

supplier   An organization that supplies materials, goods or services directly or indirectly for a customer.

system requirements   The conditions or capabilities that must be met or possessed by a system or system component to satisfy a condition or capability needed by a user to solve a problem.

test case   Each test must have a specification that contains information to identify the test, test environment, test procedure, and expected test results with acceptance criteria. An automated test will typically capture this information in the script.

test plan   A description of the technical and management approach to be followed for testing a system or component. Typical contents identify the items to be tested, features to be tested, any risks requiring mitigation, tasks to be performed, responsibilities, schedules, required resources for the testing activity, and reference to test cases. The plan must identify the types of tests that will be conducted as well as any additional tests that are needed to provide confidence that the software product does not contain any defects and to demonstrate that requirements are met.

test results   Output generated as a consequence of executing test cases. Examples of test results include logs generated manually or by automated scripts, issues identified during test and evaluation activities, test and evaluation summary report describing if/how activities deviated from the plan, summarizing results, and providing recommendations. An important element of test results is that each test case maps to its corresponding test output and that the date and time are recorded.

third party product   A third party product is an application or library used or required by a SNL ASC code application; however, ASC project teams do not normally maintain this particular software. Many of these third party product sets are developed at Sandia while other sets are developed by other government labs, commercial vendors, and university partners.

traceability   (1) The degree to which a relationship can be established between two or more artifacts of the product lifecycle, especially artifacts having a predecessor (successor or master) subordinate relationship to each other. (2) Ability to trace history, application, or location of that which is under consideration.

traceability analysis   Evaluation to determine support of the requirements for each critical artifact.

training   Activities that include specialized instruction and practice with the identified purpose of making one proficient in a skill or discipline.

trigger   Indicator that a risk has occurred or is about to occur.

unintended consequences   A principle acknowledging that human actions have at least one unforeseen outcome. This principle applies to policies, processes, work instructions, and metrics. For example, software metrics reporting too closely on individual performance (such as lines of code per unit time for each developer or number of errors in each developer's modules) frequently result in some developers "tricking" the system to achieve satisfactory performance results. These tricks may create serious quality problems and skew the results of the metrics.

user   The person or persons who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).

user support   The assistance, training, and documentation a project provides to users of its software products in ensuring that the user is satisfied and able to use the product as intended. Typically, a support plan is drawn up to specify what will be, and what will not be, provided by the project team and for what period of time.

validation   (1) Demonstrates that the product, as provided, fulfills its intended use. Validation assures "you built the right thing". (2) The process of evaluating the mathematical formulation to ensure that it adequately describes the problem of interest, that is, that the computer simulation adequately represents the real world.

verification   Addresses whether the work product properly reflects the specified requirements. Verification assures "you built it right".

Acronyms

AL	Albuquerque Office (of DOE)
AQMC	ASC Quality Management Council
ASCI	Accelerated Strategic Computing Initiative
ASC	Advanced Simulation and Computing
ASQE	ASC Software Quality Engineering
CCB	configuration change (or control) board
CM	configuration management
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CPM	Critical Path Method
CPR	Corporate Process Requirement
DMAIC	Define, Measure, Analyze, Improve, Control
DOE	Department of Energy
DP	Defense Programs
LOF	level of formality
GP&G	*ASCI Software Quality Engineering: Goals, Principles, and Guidelines*
IDEAL	Initiate, Diagnose, Establish, Act, Learn
ISO	International Organization for Standardization
MS	Microsoft
NNSA	National Nuclear Security Agency
QC-1	*DOE/AL Quality Criteria (QC-1)*
PDCA	Plan, Do, Check, Act
PERT	Program Evaluation and Review Technique
PI	principal investigator
QFD	quality function deployment
R&D	research and development
SNL	Sandia National Laboratories
SCM	software configuration management
SEPR	Simulation Enabled Product Realization
SQA	software quality assurance
SQE	software quality engineering
SSP	Stockpile Stewardship Program
SW	software
UML	Unified Modeling Language
V&V	Verification and Validation
WBS	work breakdown structure

Appendix B　　　Summary of Practices and Artifacts

The following two tables provide lists of the practices (Table 7) and artifacts (Table 8) without the descriptive details given in the practice tables.

Table 7. Software Quality Plan Practices.

Practice Number	Description of Practice
PR1	Document and maintain a strategic plan.
PR2	Perform a risk-based assessment, determine level of formality and applicable practices, and obtain approvals.
PR3	Document lifecycle processes and their interdependences, and obtain approvals.
PR4	Define, collect, and monitor appropriate process metrics.
PR5	Periodically evaluate quality problems and implement process improvements.
PR6	Identify stakeholders and other requirements sources.
PR7	Gather and manage stakeholders' expectations and requirements.
PR8	Derive, negotiate, manage, and trace requirements.
PR9	Identify and analyze risk events.
PR10	Define, monitor, and implement the risk response.
PR11	Create and manage the project plan.
PR12	Track project performance versus project plan and implement needed (corrective) actions.
PR13	Communicate and review design.
PR14	Create required software and product documentation.
PR15	Identify and track third party software products and follow applicable agreements.
PR16	Identify, accept ownership, and manage assimilation of other software products.
PR17	Perform version control of identified software product artifacts.
PR18	Record and track issues associated with the software product.
PR19	Ensure backup and disaster recovery of software product artifacts.
PR20	Plan and generate the release package.
PR21	Certify that the software product (code and its related artifacts) is ready for release and distribution.
PR22	Distribute release to customers.
PR23	Define and implement a customer support plan.
PR24	Implement the training identified in the customer support plan.
PR25	Evaluate customer feedback to determine customer satisfaction.
PR26	Develop and maintain a software verification plan.
PR27	Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.
PR28	Conduct independent technical reviews to evaluate adequacy with respect to requirements.
PR29	Determine project team training needs to fulfill assigned roles and responsibilities.
PR30	Track training undertaken by project team.

Table 8. Software Quality Plan Artifacts.

Artifact Number	Description of Artifact
AR1	Strategic plan: [project's mission, management, stakeholders]
AR2	Approved level of formality and applicable practices
AR3	Approved project processes
AR4	Process and product metrics
AR5	Project process improvement actions
AR6	Product expectations and requirements
AR7	Software requirements and attributes
AR8	List of stakeholders and organizational commitments
AR9	Project plan: [risks events, risk plan, overview, milestones, task list, resource information, roles and responsibility assignments, assumptions, constraints, dependencies, budget, schedule, SCM plan, etc.]
AR10	Project reviews and needed (corrective) actions: [risk responses, tracking and oversight responses]
AR11	Design artifacts: [documentation and/or reviews]
AR12	Implementation artifacts: [software code, assimilated other software, design documents, user documentation, developer's guide, installation guide, theory manual, interface manual, etc.]
AR13	Identification and acquisition records
AR14	Version controlled records, including baselines and associated configurations
AR15	Backup records and recovery test results
AR16	Managed issues: [product quality results (for example, non-conformances), enhancements, defects, questions, inquiries]
AR17	Release specification
AR18	Product release package (bill of materials, release notes, certification, software, etc.)
AR19	Customer support plan including training
AR20	Customer training records
AR21	Customer satisfaction evaluation
AR22	Software verification plan
AR23	Test artifacts: [test cases, test results]
AR24	Technical reviews (evidence that review occurred and review results)
AR25	Project team training needs
AR26	Project team training records

Appendix C   Mappings from Software Quality Plan to Original ASCI Applications and S&CS Practices

Original ASCI Applications and S&CS Practices	ASC Software Quality Engineering Practices
Software Engineering	
1. Requirements Phase	
1a. Gather user requirements.	PR7. Gather and manage stakeholders' expectations and requirements.
1b. Derive software requirements.	PR8. Derive, negotiate, manage, and trace requirements.
1c. Document software requirements.	PR7. Gather and manage stakeholders' expectations and requirements. PR8. Derive, negotiate, manage, and trace requirements.
1d. Assess feasibility, if applicable, and generate estimates for budget, resources, etc.	PR8. Derive, negotiate, manage, and trace requirements.
1e. Establish acceptance criteria based on requirements.	PR7. Gather and manage stakeholders' expectations and requirements.
1f. Determine necessary links to other layers of requirements, code, and tests.	PR8. Derive, negotiate, manage, and trace requirements.
1g. Ensure requirements traceability to other product artifacts throughout subsequent software phases.	PR8. Derive, negotiate, manage, and trace requirements.
1h. Review and approve requirements artifacts.	PR8. Derive, negotiate, manage, and trace requirements. PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
2. Development: Design Subphase	
2a. Derive the design.	PR13. Communicate and review design.
2b. Communicate the design to the team.	PR13. Communicate and review design.
2c. Document the design.	PR13. Communicate and review design.
2d. Evaluate impact to requirements.	PR13. Communicate and review design. PR8. Derive, negotiate, manage, and trace requirements.
2e. Plan for testing: initiate development of test plan.	PR13. Communicate and review design. PR26. Develop and maintain a software verification plan.
2f. Review and approve design artifacts.	PR13. Communicate and review design PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
3. Development: Implementation Subphase	
3a. Evaluate impact of implementation to design and requirements.	PR13. Communicate and review design. PR14. Create required software and product documentation. PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
3b. Translate design into code and other software product artifacts.	PR13. Communicate and review design. PR14. Create required software and documentation.
3c. Communicate issues with requirements/design team and developers.	PR13. Communicate and review design. PR14. Create required software and documentation.
3d. Review and approve implementation artifacts.	PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.

Original ASCI Applications and S&CS Practices	ASC Software Quality Engineering Practices
4. Development: Test Subphase	
4a. Finalize test plan.	PR26. Develop and maintain a software verification plan.
4b. Execute test cases found in test plan.	PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.
4c. Review test case output using acceptance criteria defined in test plan.	PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
4d. Document test case results.	PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.
4e. Retest updated software if acceptance criteria are not satisfied.	PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.
4f. Review and approve Test Subphase outputs.	PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
5. Release Phase	
5a. Receive and evaluate release request.	PR20. Plan and generate the release package.
5b. Plan and develop release.	PR20. Plan and generate the release package.
5c. Review and approve release.	PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution. PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.
5d. Create and distribute release.	PR20. Plan and generate the release package. PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution. PR22. Distribute release to customers.
5e. Support release, as agreed with customer.	PR23. Define and implement a customer support plan. PR24. Implement the training identified in the customer support plan. PR25. Evaluate customer feedback to determine customer satisfaction.
Project Management	
6. Project Planning	
6a. Submit IP addressing project tasks annually.	PR11. Create and manage the project plan.
7. Tracking and Oversight	
7a. Review milestone status quarterly.	PR12. Track project performance versus project plan and implement needed (corrective) actions.
7b. Issue Baseline Change Proposals (BCPs), if needed.	PR12. Track project performance versus project plan and implement needed (corrective) actions.
7c. Prepare performance reports on a quarterly basis.	PR12. Track project performance versus project plan and implement needed (corrective) actions.
8. Risk Management	
8a. Incorporate risk identification and risk mitigation into project execution using the BCP.	PR9. Identify and analyze risk events. PR10. Define, monitor, and implement the risk response.

Original ASCI Applications and S&CS Practices	ASC Software Quality Engineering Practices
Support Elements	
9. Requirements Management	
9a. Conduct requirements tracing.	PR8. Derive, negotiate, manage, and trace requirements.
9b. Determine requirements ownership and status tracking.	PR8. Derive, negotiate, manage, and trace requirements.
10. Configuration Management	
10a. Conduct issue tracking of software product artifacts, including requirements.	PR18. Record and track issues associated with the software product.
10b. Perform version control of software product artifacts, including requirements.	PR17. Perform version control of identified software product artifacts.
10c. Perform release and distribution management.	PR20. Plan and generate the release package. PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution. PR22. Distribute release to customers.
10d. Engage in ASCI records management.	
11. Third Party Software	
11a. Accept third party software and libraries into the application code domain.	PR15. Identify and track third party software products and follow applicable agreements. PR16. Identify, accept ownership, and manage assimilation of other software products.
11b. Install, integrate, & control the accepted third party software.	PR15. Identify and track third party software products and follow applicable agreements. PR16. Identify, accept ownership, and manage assimilation of other software products.
12. Training	
12a. Train appropriate project members in use of project management and project tracking and oversight processes.	PR29. Determine project team training needs to fulfill assigned roles and responsibilities. PR30. Track training undertaken by project team.
12b. Train staff on activities necessary for producing software artifacts.	PR29. Determine project team training needs to fulfill assigned roles and responsibilities. PR30. Track training undertaken by project team.
12c. Train staff on use of software tools.	PR29. Determine project team training needs to fulfill assigned roles and responsibilities. PR30. Track training undertaken by project team.
12d. Train staff on software processes and their implementation.	PR29. Determine project team training needs to fulfill assigned roles and responsibilities. PR30. Track training undertaken by project team.
12e. Train staff on software verification process and techniques.	PR29. Determine project team training needs to fulfill assigned roles and responsibilities. PR30. Track training undertaken by project team.

Appendix D   Template for an Assessment Checklist

This appendix includes an assessment checklist based on the practices and suggested artifacts of this document. Periodically, ASC management will review this checklist and modify it as necessary. Different ASC program elements may choose to tailor the checklist to best suit the needs and goals of that program element. One program element may decide to evaluate the overall effectiveness of each practice as a single score. Another program element may determine that providing two scores, one for approach and another for results, provides better assessment information and feedback. An assessment checklist identifies practices and may indicate assessment goals; the checklist can be used by an independent assessment team or by a project team for self-assessment. The assessment criteria used by an independent team will be communicated to software projects scheduled for assessment prior to the start of the actual assessment.

See section 5 for a discussion of the assessment strategy of this Software Quality Plan**.**

D.1   Instructions for Completing Assessment Checklist

The details of the activities that comprise each practice are not listed separately in the Assessment Checklist that is in section D.2. Listing all of the required test types that should be included in the test plan and then subsequently executed would result in a checklist that is unwieldy. The Project Team Evaluation below provides a set of guidelines for assessment of practice implementation.

Definitions of the columns in the Assessment Checklist are provided below.

(1)   Project Team Name/Assessment Date
   This column includes the name of the ASC software project and the date of the assessment.

(2)   Project Team Evaluation
   This is the column the software project team fills in to determine where they are in terms of performing or implementing all recommended practices. A code team will select a value between 0-5 or "NA" based on the criteria specified below. The assessment values discussed in this section are suggestions only. In previous ASC program level assessments a scale of 1 to 3 has also been used. At the beginning of an assessment period the assessment sponsor working with the independent assessment team will establish the appropriate assessment values that will be consistent in accomplishing the assessment goals and objectives.

   5   *Outstanding* – the software project team has fully implemented this practice. This is the most difficult value to achieve. This value indicates that the practice is at the maintenance stage. Evidence exists that the practice is integrated into the project team's development process. Concurrence by the assessment team is needed for the practice to be officially recognized as fully implemented. To be at the fully implemented level, a documented process for the practice needs to exist, all team members are fully trained on the process, work products have been produced and deemed by the assessment team to be reproducible, and practice plans and results have been shared with all appropriate stakeholders. The project is 'outstanding' in its implementation of this practice.

   4   *Complete* – the software project team has implemented a final (not draft) process and work products are in place supporting this practice. Most project team members have been trained in the process implementation. Practice results have been shared with some stakeholders. Everything is in place for this practice to become rated at a '5' but there are still a few activities that need to be addressed (for example, training, reproducing work products, or sharing results with stakeholders). The project is 'complete' in this area but not yet 'outstanding'.

3 *Good* – the software project team has partially implemented this practice. Some evidence exists that the practice has started. Resources for the fulfillment of this practice have been identified, but the implementation is not complete. For example, a draft of the process for conducting the practice exists or a completed documented process exists with most of the team (but not all) complying with the process. There is evidence of significant progress on rolling out an implementation for the process. Evidence also exists of draft work products that contain significant content. Additional resources most likely will be needed to raise this practice to 'complete' or 'outstanding.'

2 *Fair* – the software project team has preliminary evidence for implementing this practice. There may be a preliminary plan for how they will proceed with a process and its implementation and preliminary work products may exist. Much work is needed to move toward a 'complete' or 'outstanding' rating on this practice.

1 *Limited* –the software project team has proposed an implementation of this practice. At this level, it is typical that resources have not yet been identified and allocated for fulfillment of the practice. Activities and resources for the practice are in the planning stages but some evidence exists that the project is committed to implementing this practice.

0 *Absent* – the software project team has not yet addressed the implementation of this practice.

NA The software project team determines this practice is not applicable to its code development environment. A value of NA must be accompanied by an explanation from the team describing why the practice will not be followed.

Note: Specific guidelines for selecting assessment values will be provided by ASC management for each entry in the Assessment Checklist. If the ASC management recommendation for a particular practice, such as practice **PR8**: "Derive, negotiate, manage and trace requirements," is five then the expectation is that all activities addressed in the description of that practice will be carried out in order for a code team to achieve a value of '5' in its self-assessment.

(3) Assessment Team Evaluation
As needed, ASC management will appoint an independent assessment team to review the current state of practices performed by each team. The independent assessment team will use the same scale as the project team [see (2) above].

(4) Comments for Project Team or Assessment Team
This column is intended to record comments about a project team's particular implementation of a given practice or why that practice is not applicable. The column will also be used to record evidence of implementation of that practice, especially to show 'outstanding', 'complete', or 'good' implementation. Either the software project team or the assessment team may enter information in this column. The author of the comment should be clearly identifiable.

(5) Completed By
This line indicates the person (project team, assessment team) who completed the assessment checklist. The person who signs this section should print their name, date the checklist, and add their signature.

Software project teams should use a tool as directed by ASC management to determine how closely they are adhering to the ASC Software Quality Plan. In addition to highlighting areas that are appropriate for increased improvement efforts, the software project teams can observe how they are improving by comparing the scores of various practices from one assessment period to the next.

D.2   Assessment Checklist for ASC Software Areas

(1) Project Team Name: Assessment Date:	(2) Project Team Evaluation	(3) Assessment Team Evaluation	(4) Comments for Project Team or Assessment Team
Practice	5 = Outstanding 4 = Complete 3 = Good 2 = Fair 1 = Limited 0 = Not addressed NA – not applicable	5 = Outstanding 4 = Complete 3 = Good 2 = Fair 1 = Limited 0 = Not addressed NA – not applicable	Use this area to explain why NA is selected as a response to columns (2) or (3) and to demonstrate evidence for other responses as needed.
Project Management (12)			
1. Strategic Planning			
PR1. Document and maintain a strategic plan.			
2. Determination of Applicable Practices and Level of Formality			
PR2. Perform a risk-based assessment, determine level of formality and applicable practices, and obtain approvals.			
3. Process Implementation and Improvement			
PR3. Document lifecycle processes and their interdependencies, and obtain approvals.			
PR4. Define, collect, and monitor appropriate process metrics.			
PR5. Periodically evaluate quality problems and implement process improvements.			
4. Requirements Engineering			
PR6. Identify stakeholders and other requirements sources.			
PR7. Gather and manage stakeholders' expectations and requirements.			
PR8. Derive, negotiate, manage, and trace requirements.			
5. Risk Management			
PR9. Identify and analyze risk events.			
PR10. Define, monitor, and implement the risk response.			
6. Project Planning, Tracking, and Oversight			
PR11. Create and manage the project plan.			
PR12. Track project performance versus project plan and implement needed (corrective) actions.			
Software Engineering (13)			
7. Software Development			
PR13. Communicate and review design.			
PR14. Create required software and product documentation.			
8. Integration of Third Party or Other Software			
PR15. Identify and track third party software products and follow applicable agreements.			
PR16. Identify, accept ownership, and manage assimilation of other software products.			

(1) Project Team Name: Assessment Date:	(2) Project Team Evaluation	(3) Assessment Team Evaluation	(4) Comments for Project Team or Assessment Team
9. Configuration Management			
PR17. Perform version control of identified software product artifacts.			
PR18. Record and track issues associated with the software product.			
PR19. Ensure backup and disaster recovery of software product artifacts.			
10. Release and Distribution Management			
PR20. Plan and generate the release.			
PR21. Certify that the software product (code and its related artifacts) is ready for release and distribution.			
PR22. Distribute release to customers.			
11. Customer Support			
PR23. Define and implement a customer support plan.			
PR24. Implement the training identified in the customer support plan.			
PR25. Evaluate customer feedback to determine customer satisfaction.			
Software Verification (3)			
12. Software Verification			
PR26. Develop and maintain a software verification plan.			
PR27. Conduct tests to demonstrate that acceptance criteria are met and to ensure that previously tested capabilities continue to perform as expected.			
PR28. Conduct independent technical reviews to evaluate adequacy with respect to requirements.			
Training (2)			
13. Training			
PR29. Determine project team training needed to fulfill assigned roles and responsibilities.			
PR30. Track training undertaken by project team.			
Total Number of Areas	**13**		
Total Number of Practices	**30**		
(5) Completed By: (print name and date) (signature)	--------------------------- --------------- ---------------------------		

Appendix E   Test Categories

The following definitions for *test case, test plan,* and *test results* are included to provide context for the project teams as they develop and maintain their software verification plan and conduct necessary testing throughout the product lifecycle.

Test case   Each test must have a specification that contains information to identify the test, test environment, test procedure, and expected test results with acceptance criteria. An automated test will typically capture this information in the script.

Test plan   A description of the technical and management approach to be followed for testing a system or component. Typical contents identify the items to be tested, features to be tested, any risks requiring mitigation, tasks to be performed, responsibilities, schedules, required resources for the testing activity, and reference to test cases. The plan must identify the types of tests that will be conducted, as well as any additional tests that are needed to provide confidence that the software product does not contain any defects and to demonstrate that requirements are met.

Test results   Output generated as a consequence of executing test cases. Examples of test results include logs generated manually or by automated scripts, issues identified during test and evaluation activities, test and evaluation summary report describing if/how activities deviated from the plan, summarizing results, and providing recommendations. An important element of test results is that each test case maps to its corresponding test output and that the date and time are recorded.

The following kinds of tests should be tailored for appropriate coverage according to the level of formality implied by the risk-based analysis. Test harnesses, scripting languages including languages such as Expect, and automated test generation tools can help support the following kinds of tests.

(a) *General testing* covers tests that need to be conducted on all software products to meet specific requirements: code coverage, memory testing, and static testing.

- *Code statement coverage*: Evidence must be provided demonstrating that the requisite percentage of the software source statements related to essential requirements associated with the code's intended use has been executed through testing. The developer is key in determining which code implements essential requirements. Applying an automated tool that uses a specified set of tests, such as the regression tests, typically provides this evidence. An automated coverage analysis tool is very useful in checking code coverage.

- *Static Testing:* Static testing includes the checking provided during compilation and other static code analysis tools, such as lint and flint.

- *Memory and Resource Leak Testing:* This type of testing is a white-box testing methodology used to determine that the program is properly using memory and not generating any other resource leaks, such as file descriptors and scratch files. Memory testing is programming-language dependent. Commercial memory and resource leak detection tools, such as Purify and Insure++, support this type of testing.

(b) *Unit tests* are developed, maintained, and performed on code units with respect to their requirements, specifications, and design during the development lifecycle. Typically conducted prior to integration testing, unit testing is the process of testing the individual units or modules of a program before they are integrated into the software product.

(c) *Integration tests* involve testing part or all of the system to evaluate the interactions among components. For example, third party software capabilities that the software project relies on (or could rely on) should be tested alongside the software components that use those capabilities.

(d) **Regression tests** are developed, maintained, and performed to check that code modifications have not introduced unintended effects, the code works as expected for all computational platforms supported, and that the code still meets its specified requirements.

(e) **User acceptance tests** are performed to determine that the software system to be delivered is adequate for its intended use by the user community. This testing, if performed primarily by a code team rather than customers, could also be termed user *perspective* testing.

(f) **System software tests** use a method or combination of methods to ensure that required functional features satisfy specified requirements.

(g) **Installation tests** are required for released software on all required target platforms. This testing seeks to confirm that the software installation on the target platform occurred correctly. Installation tests are useful as installation routines are often the most heavily modified part of the product.

A subset of test cases previously developed can be used with additional tests designed specifically for the process of installation. This type of testing typically occurs during the release activities. Typically, installation tests are delivered with the software for the end user to execute and compare to expected results. Installation tests must address:

- that the variety of options and combinations of options selected by the user were acceptable
- that the installation was performed on an approved hardware configuration
- that required interconnections to other programs were properly established.

Appendix F   Techniques and Tools

ASQE Area	Techniques	Tools
Process Implementation	• PDCA (Plan, Do, Check, Act) • IDEAL (Initiate, Diagnose, Establish, Act, Learn) • DMAIC (Define, Measure, Analyze, Improve, Control) Lifecycle models: • Iterative • Spiral • Sequential Software development methodologies: • Agile • RUP (Rational Unified Process) • Waterfall	• Rational Tool Suite
Process Improvement	Metric-based techniques for process improvement include: • Collecting data • Root cause analysis • Statistical process control • Design of experiments to improve robustness in parameters, products, and processes • PDCA (Plan, Do, Check, Act) • IDEAL (Initiate, Diagnose, Establish, Act, Learn) • DMAIC (Define, Measure, Analyze, Improve, Control) Non-metric techniques for process improvement include: • Error-proofing and preventive actions • Improving process definitions and their associated documentation • Corrective actions • Benchmarking • Peer and management reviews • Implementation of improvement suggestions Metric-based techniques are generally more effective than non-metric techniques in effecting process improvement. However, the metric-based techniques require an understanding of the appropriate statistical analyses, process variation, and "unintended consequences" of the metrics.	• Scatter diagrams • Histograms • Check sheets • Pareto analysis • Cause and effect diagrams • Control charts
Requirements Engineering	• Derivation techniques Techniques to gather and analyze requirements include: • creating prototypes • graphical models (context diagrams, use cases, information models, state-transition diagrams) • quality function deployment (QFD) that relates product features and attributes to customer value. • Requirements negotiations with users/stakeholder	• Requirements management tools (for example, DOORS and ReqPro, or other less automated tools like Excel and Word) • Configuration control board (CCB) for reviewing, analyzing, and determining the disposition of proposed changes to baselined requirements

ASQE Area	Techniques	Tools
Risk Management	• Risk identification techniques (for example, checklist, taxonomy, Delphi) • Risk analysis techniques (for example, expert judgment, simulation, decision management approach, monitoring trees) • Risk approach	• Risk management tool for storing and tracking the project risks (for example, Risk Radar, Excel) • Monitoring tool
Project Planning	Project planning approach	• Planning tools and templates • Task evaluation tools • Work breakdown structure (WBS) • Gantt charts, • PERT charts • CPM charts
Tracking & Oversight	• Performance based review approaches • Negotiations with management and stakeholders	• Task management tools
Configuration Management	• SCM plan specifying project standards, file naming conventions, and SCM project responsibilities	• Version control tools (for example, CVS, PVCS VM, PVCS Dimensions, ClearCase) • Issue tracking tools (for example, SourceForge, PVCS Dimensions, ClearQuest)
Release & Distribution Management	• Release and distribution approach or plan	• Build tools
Customer Support	• Training on topics such as negotiation strategies, social styles, customer satisfaction, customer service, etc.	• Customer support survey • Service level agreement
Software Verification	• Pair programming • Peer reviews • Prioritization of test and evaluation activities	• Coverage analysis tool • Static code analysis tool • Memory testing tool • Test harness • Automated test generation tool
Training	• Product tutorials • Classroom training • Web-based training	• Student evaluations of training classes • Training matrix identifying roles, responsibilities, and necessary skills
Metrics	• Goal/question/metric paradigm • Cause and effect diagram	

Appendix G   SNL Practices as an Implementation of the GP&G SQE Guidelines

The following chart maps between the *ASCI Software Quality Engineering: Goals, Principles and Guidelines*, a report developed collaboratively as high-level SQE guidelines for software developed in the Tri-Laboratory ASCI Program, and the SNL site-specific practices described in this document.

GP&G SQE Guidelines		SNL Practices
Software Verification	**Technical reviews** • Technical soundness • Static analysis	PR26, PR28 PR26, PR28
	Unit testing • Traceable, repeatable component tests	PR26, PR27, PR17, PR18
	Regression testing • Building the code • Executing tests • Feature-based test suite for multiple platforms	PR17, PR18, PR19, PR20, PR21 PR27 PR26
	Comparison techniques • Analytic solutions • Other codes' results	PR26, PR27 PR26, PR27
	User acceptance testing • Applicability evaluation • Usability evaluation • Code confidence • Results credibility	PR7, PR8, PR13, PR14, PR20, PR26, PR27, App. E, Table 2, PR28, App. E PR20, PR27, App. E Goals of SQ Plan, PR20, PR27, App. E, App. F Goals of SQ Plan, PR20, PR27, App. E, App. F
	Training • Verification methods and techniques	PR29, PR30, Table 1

Software Engineering	**Lifecycle management**
	• Time-based work flow
	• Requirements
	  Design
	  Construction
	  Test
	  Support activities
		PR3
		PR6, PR7, PR8
		PR13
		PR14
		PR23, PR24
		PR8, PR15, PR16, PR17, PR18, PR19, PR22, PR23, PR24, PR25

	Configuration management • Version management • Issue tracking • Release management	PR15, PR16, PR17 PR18 PR17, PR18, PR19
	Measurements and metrics • Software products • Software processes	PR4, all practices suggest metrics PR4, all practices suggest metrics
	Reviews and assessments • Management reviews • Technical reviews	PR12, section 5, App. D PR27, section 5, App. D

GP&G SQE Guidelines		SNL Practices
	Process improvement • Engineering process baseline • Identified improvements • Improvement implementation	PR3 PR5 PR5, PR18, PR21
	Training • Software practice methods and techniques	PR29, PR30, Table 1
Project Management	**Risk management** • Risk assessment • Risk control	PR9 PR10
	Requirements management • Gathering, documenting, verifying, managing change to requirements	PR6, PR7, PR8 PR26, PR27, PR28 PR18
	Project planning • Statement of work • Constraints and goals • Implementation plan • Resource assessment	PR11 PR11 PR11 PR11
	Tracking and oversight • Actual results vs. planned results • Corrective action	PR12 PR12
	Process management • Process documentation and plans • Technology improvement • Improvement leverage	Table 1, PR3 PR4, section 5, Table 1 PR4, section 5, Table 1
	Training • Project management methods and techniques	PR29, PR30, Table 1

Distribution

External Distribution:

Department of Energy
1000 Independence Avenue SW
Washington, DC 20585
Attn: Jamileh Soudah, DP-51 (1)
         Njema Frazier, DP-51 (1)
         Dimitri Kusnezov, DP-51 (1)

Los Alamos National Laboratory
Mail Station 5000
P.O. Box 1663
Los Alamos, NM 87545
Attn: M. Peterson (1)
         L. Cox (1)
         K. Koch, MS F652 (1)
         J. LaGrange, MS D445 (1)
         James Peery (1)
         Scott Doebling (1)
         Jim Morel (1)

University of California
Lawrence Livermore National Laboratory
7000 East Avenue
P.O. Box 808
Livermore, CA 94550
Attn: E. Dube, MS L-095 (1)
         Roger Logan, MS L-125 (1)
         C. Nitta, MS L-096 (1)
         N. Storch, MS L-303 (1)
         Tom Adams (1)

SNL Distribution:

Copies	Mail Stop	Name, Organization	Copies	Mail Stop	Name, Organization
2	MS0139	Ann Hodges, 9905	2	MS0376	Bob Kerr, 9226
2	MS0376	Ted Blacker, 9226	2	MS0378	Richard Drake, 9231
2	MS0382	Edward Boucheron, 9141	2	MS0382	H. Carter Edwards, 9143
10	MS0660	Molly Ellis, 9515	10	MS0660	Christi Forsythe, 9515
2	MS0660	Patricia Hackney, 9515	2	MS0661	Joseph Schofield, 9516
2	MS0817	Sue Kelly, 9224	2	MS0822	Constantine Pavlakos, 9326
2	MS0822	Judy Sturtevant, 9326	2	MS1111	Robert Heaphy, 9215
10	MS1138	C. Michael Williamson, 6225			

Housekeeping Distribution:

1	MS9018	Central Technical Files, 8945-1
2	MS0899	Technical Library, 9616