SANDIA REPORT
SAND2002-0121
Unlimited Release
Printed January 2002

ASCI Applications Software Quality Engineering Practices

John Zepper, Kathy Aragon, Molly Ellis, Kathleen Byle, and Donna Eaton

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2002-0121
Unlimited Release
Printed January 2002

Sandia National Laboratories ASCI Applications Software Quality Engineering Practices

Version 1.0

John Zepper and Kathy Aragon
Production Computing/SIERRA Architecture

Molly Ellis, Kathleen Byle, and Donna Eaton
Information Technology and Data Modeling

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

Abstract

This document provides a guide to the deployment of the software verification activities, software engineering practices, and project management principles that guide the development of Accelerated Strategic Computing Initiative (ASCI) applications software at Sandia National Laboratories (Sandia). The goal of this document is to identify practices and activities that will foster the development of reliable and trusted products produced by the ASCI Applications program. Document contents include an explanation of the structure and purpose of the ASCI Quality Management Council, an overview of the software development lifecycle, an outline of the practices and activities that should be followed, and an assessment tool. These sections map practices and activities at Sandia to the *ASCI Software Quality Engineering: Goals, Principles, and Guidelines*, a Department of Energy document.

Acknowledgements

The authors would like to thank the following individuals for their reviews, comments, and contributions in preparing this document: Henry Abeyta, Dan Carroll, Edward Cull, David Cuyler, Carter Edwards, Joe Fernandez, Christi Forsythe, Gary Froehlich, Ann Hodges, Scott Hutchinson, Stephen Lott, Mike McGlaun, David Peercy, James Peery, Martin Pilch, Harold Radloff, Rhonda Reinert, Alex Treadway, Janice Washington, William Moffatt, David Womble, and all code team members and managers who reviewed this document.

Table of Contents

List of Figures

List of Tables

Executive Summary

This document is the Sandia National Laboratories (Sandia) Applications program deployment of the Department of Energy (DOE) document *ASCI Software Quality Engineering: Goals, Principles, and Guidelines* (GP&G). The GP&G specifies the Accelerated Strategic Computing Initiative (ASCI) program's requirements for software quality engineering at each laboratory. This document and the GP&G both map to *Quality Criteria (QC-1)*, a document produced by the Department of Energy/Albuquerque Office (DOE/AL). Both Sandia's document and the GP&G recognize the significance of following the QC-1 standard in the development of nuclear weapons software codes.

This document builds on the GP&G foundation to specify tangible practices and activities that will establish confidence in our codes and credibility in our results. The document includes the following:

- tailored GP&G requirements to fit the software development process of the applications program
- a description of management involvement in the software quality improvement process
- a description of the software quality improvement process

This document establishes the application code teams' commitment to improving their software products by applying cost-effective software engineering quality practices. These practices comprise an important part of the ASCI Verification and Validation Program. Individuals interested in validation issues should contact the Verification and Validation program, which is responsible for validation of the models. Those interested in the overall structure of the ASCI program and the interplay of its parts should consult the ASCI Program Plan or ASCI Implementation Plans.

This document is organized into four sections. Section 1 describes how Sandia has integrated the GP&G requirements into the NNSA (National Nuclear Security Agency) ASCI program. Section 2 discusses the ASCI Quality Management Council (AQMC). The council's purpose is to sustain and improve software process and products throughout the defined lifecycle. Section 3 enumerates the main practices that compose the development of Sandia ASCI application software. Section 4 presents an assessment tool that was developed based on the practices in this document. This tool provides application code teams a method for performing a self-assessment and gap analysis. Information produced by using the tool will enable application code teams and management to perform path-forward analysis for software process improvement. Appendix A defines special terms and acronyms. Appendix B illustrates how this document maps and aligns with the GP&G as well as to QC-1. Appendix C consists of a blank assessment tool.

Commitment

The Sandia ASCI Applications program will follow the processes, practices, and activities outlined in this document. Thus the project teams will provide accountability to NNSA in demonstrating consistent SQE results. The goal of this document is to foster organizational consistency by defining common practices and by facilitating the use of common tools and processes where feasible. These practices and activities will be modified and improved as the code development process matures. Our intent is to provide tangible evidence demonstrating high confidence in ASCI simulations at Sandia.

Approved By

_____ _____

Michael McGlaun, Date
Sandia ASCI Applications Manager

Concurred By

_____ _____

Thomas Bickel, Date
Sandia Director,
Engineering Sciences Center

_____ _____

Michael Vahle, Date
Sandia ASCI Program Manager

_____ _____

Henry Abeyta,
Deputy Director for System Engineering, Date
Sandia/NM

_____ _____

Edward Cull,
Deputy Director for Weapons System Engineering, Date
Sandia/CA

1 Introduction

1.1 Background

The National Nuclear Security Agency (NNSA) has created 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 has enabled the Accelerated Strategic Computing Initiative (ASCI) to support the SSP in transitioning from test-based to computational modeling and simulation-based methods. The ASCI program will adhere to the specifications for software quality assurance defined in the document *Quality Criteria (QC-1)* produced by the Department of Energy/Albuquerque Office (DOE/AL).

The ASCI 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 ASCI software projects. The GP&G specifies that each laboratory will select and tailor their best practices to achieve the stated goals of 1) establishing confidence in codes and 2) establishing credibility in results.

The GP&G organizes the ASCI guidelines into three major areas: 1) software engineering, 2) software verification, and 3) project management. The GP&G requires that each site develop its own specific practices to appropriately implement the guidelines. Taking direction from the GP&G, this document includes an assessment tool that provides a method of identifying the current state of site-specific practices for applications at Sandia National Laboratories (Sandia). This document also provides a mechanism for facilitating improvement of those practices.

1.2 Purpose

The purpose of this document is to describe practices that will maintain a high level of confidence in ASCI-developed software at Sandia. The document is organized to provide a straightforward guide to the deployment of the software engineering practices, verification activities, and project planning and oversight practices that guide the development of ASCI applications software at Sandia.

This document explains the purpose of the ASCI Quality Engineering Management Council (AQMC) in overseeing and improving software initiatives from an organizational perspective. The document provides an overview of the Sandia ASCI applications software-development lifecycle. This lifecycle specifies the practices that should be followed in developing robust, effective, and efficiently written applications. A checklist of recommended practices is provided in the assessment tool, and a mapping mechanism is included (in Appendix B) that traces these practices to the GP&G to satisfy the goals of that document. The practices identified herein require that individual application code teams be responsible for implementing and producing evidence that demonstrates adherence to requirements of this document. The documents and their owners are illustrated in Figure 1.

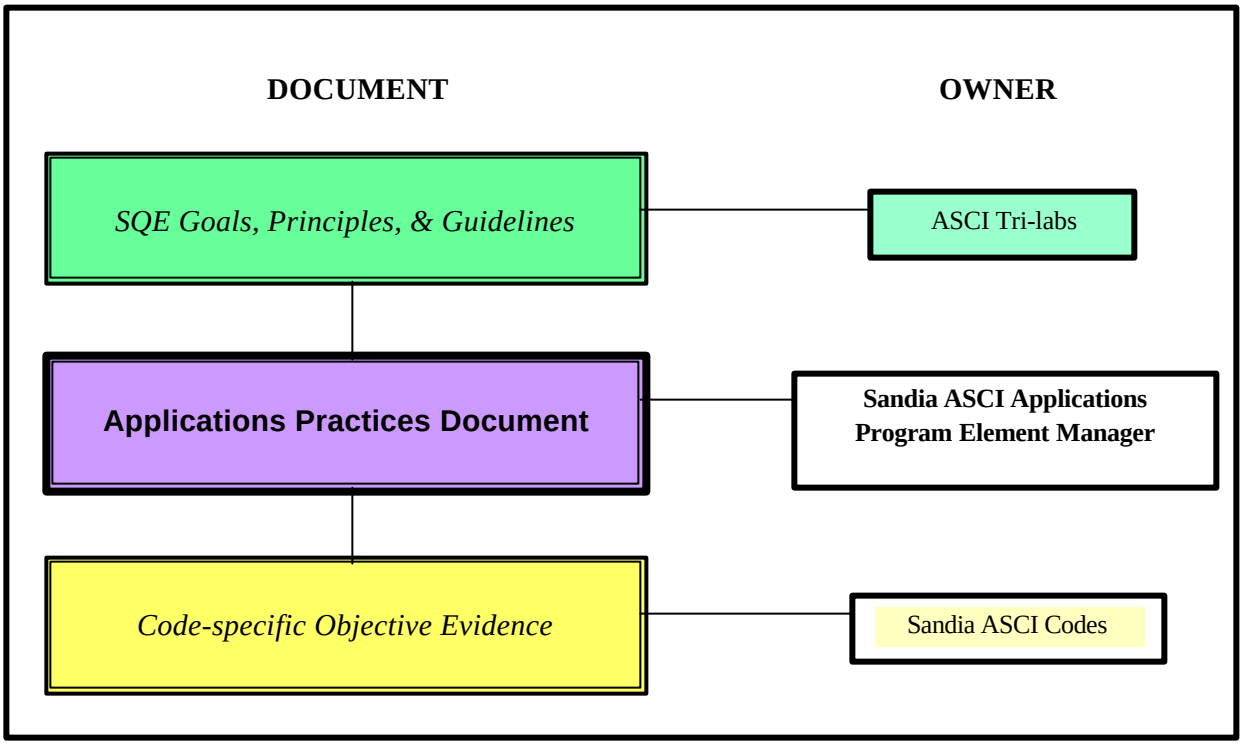

Figure 1. Context of practices document.

The following entities are responsible for direction and implementation of the documents in Figure 1:

Entity	Responsibility
NNSA	Provides guidance to the ASCI Tri-labs in developing GP&G
AQMC	Sets policy and ensures institutionalization of practices
Sandia Applications	Implements GP&G
Sandia V&V	Provides independent assessment verification and validation (V&V) of application code teams in applying GP&G
Sandia Code Teams	Compiles/maintains objective evidence

1.3 Scope

The provisions of this document pertain to the development and support of software within the Sandia ASCI Applications program. The practices that are outlined are especially intended to target ASCI application codes.

Figure 2 illustrates the context of the ASCI application codes in relationship to stockpile-driven applications.

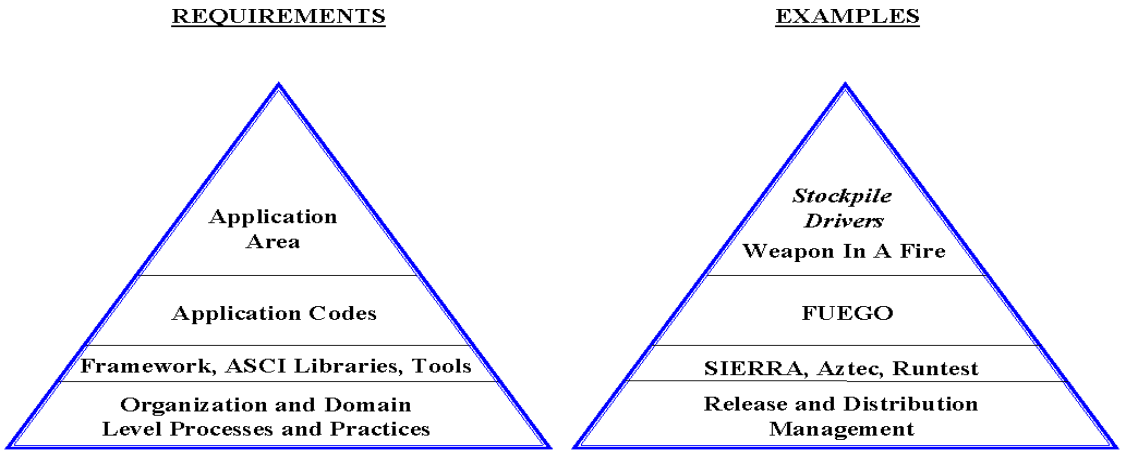

Figure 2. Requirements flow pyramid.

This document is part of an existing and planned suite of guidance and requirements documents that are intended to institutionalize traceable credibility to stockpile computing activities. These other documents are

- Guidelines for V&V Plans: *Guidelines for Sandia ASCI Verification and Validation Plans: Version 2.0,* SAND2000-3101, January 2001
- Peer Review: *Peer Review Process for the Sandia ASCI V&V Program: Version 1.0,* SAND2000-3099, January 2001

Examples of current code team applications to which this document applies are illustrated in Table 1.

Table 1. Sandia ASCI Applications

Application Category	Application Name
Framework	SIERRA, ALEGRA
Thermal, Fluid	CALORE, FUEGO, PREMO, VIPAR, ARIA
Structural and Solid Mechanics	SALINAS, PRESTO, ADAGIO, ANDANTE
Electrical Device and Circuit	HPEMS (e.g. XYCE)
ElectroQuasiStatics	ALEGRA
Shock Physics	ALEGRA
Libraries and Algorithms	Trilinos, Petra, Dakota, Verde, Zoltan, ACME
Particle Transport	ITS, CEPTRE, NuGET
Electromagnetics	EMPHASIS, CABANA
Mesh Generation	CUBIT

See Section 3.5.3 for additional detail on third party software that is used by any of the applications.

1.4 Graded Approach

Sandia ASCI applications software project teams will use a graded approach in applying the practices described in this document. A graded approach means that projects will apply a level of formality and rigor appropriate to their application. The following guidelines for determining an appropriate class apply:

- **Class A** codes will include applications intended for weapon design or qualification. All of the ASCI-funded codes listed in Table 1, plus future codes that come under the Sandia ASCI applications umbrella that are intended for weapon design or qualification, are Class A projects. All of the practices identified in the assessment tool will be required for Class A software development.

- **Class B** codes are not intended for use in weapon design or qualification. Examples include ASCI-funded research codes or prototype software that has not been incorporated into a production code. Class B projects are not required to address all of the practices in the assessment tool. They are, however, expected to demonstrate good project management practices, a clear understanding of what is expected of the software requirements, and a method of determining whether the code meets the requirements through tests and test plans.

- **Class C** codes may be used for weapon design and qualification but are not listed in Table 1. These legacy codes, not supported by ASCI, have possibly been in existence for some time and may be in a redevelopment state (being rewritten to one of the applications listed in Table 1).

By considering impact in the ASCI production environment (column 2 of Table 2), projects can identify the class for their activities. Project leads are responsible for self-assessing their class. This class must be reviewed and approved by the AQMC.

Table 2. Class Identification Tool

Class Categories		
Class	**Impact in ASCI Production Environment**	**Adherence to Practices Listed in Assessment Tool**
A	Used in weapon design or qualification	As specified by the AQMC
B	Not used in weapon design or qualification R&D code prototype system	Not all practices required, only: Project management: 6a,7a,7b,7c,8a Requirements Phase: 1a,1b,1c,1d,1e,1f,1g Test Subphase: 4a,4b,4c,4d,4e,4f
C	Existing legacy application not being developed under ASCI program auspices	Not bound by practices; should apply as appropriate

2 ASCI Quality Management Council

Implementation of the GP&G recommendations requires the commitment, support, and oversight of the organizations performing the work to ensure that software engineering process improvements are applied consistently and effectively. To fulfill the requirements of QC-1, the Sandia ASCI Applications program director has established the AQMC (ASCI Quality Management Council). The AQMC is an oversight group that is responsible for setting and directing the strategy for implementing quality systems, including software engineering processes and software process improvements, for all ASCI software projects. The AQMC will ensure consistent and cost effective implementation of software quality engineering in all ASCI software projects.

The AQMC reports to the ASCI program director and is composed of the program element managers who have software development and maintenance activities within their program element, the V&V program element manager, and the ASCI program manager. The AQMC will meet at least twice a year to review the development and implementation of software engineering practices and will publish an annual report on the state of SQE within ASCI.

Responsibilities of the AQMC include

- setting priorities
- communicating best practices among the software development teams
- monitoring and documenting compliance with guidelines set forth in this document
- authorizing modifications to policies and practices
- reviewing and assessing quality initiatives in the ASCI program
- coordinating independent and external assessments
- maintaining this document and any other documents under its purview
- convening working groups to support development of policies and practices

The AQMC will establish the software engineering practices that must be implemented on software development projects throughout the ASCI program. The AQMC will use a phased approach in establishing requirements commensurate with the stage in the lifecycle of software development efforts. The assessment tool, discussed in Section 4, will be updated and published annually as a mechanism for communicating the requirements baseline. The Sandia ASCI Applications program element manager is responsible for implementing this report's practices in the Sandia ASCI Applications program.

3 Software Quality Engineering Practices

The Sandia ASCI Applications SQE program is described in this section. This program has been developed following the organization of the GP&G: Software Verification, Software Engineering, and Project Management. A fourth concept has been added—Support Elements. Support Elements capture aspects common to all three of the guidelines, such as training, or those that have overarching implications for the success of the software program, such as configuration management. See Figure 3 for a pictorial representation of the program organization.

The organization represented in Figure 3 encompasses the main principles that guide the development of Sandia ASCI application software. The heart of this figure is the software lifecycle phases – Requirements, Development, and Release -- that include the core practices of Software Engineering. Overlying the lifecycle is Project Management and underlying it is Software Verification. Project Management provides the planning practices, while Software Verification provides the assurance practices throughout the lifecycle. Applying to every product activity are the Support Elements, which include such disciplines as requirements management, configuration management, third party software management, and training. The software lifecycle phases and associated practices will be the focus of discussion in this section; however, the other guideline areas will be explained and their significance will be briefly explored.

This document requires no strict chronology of events, provided the requirements of all the phases are satisfied, nor does it preclude the implementation of any specific development methodologies.

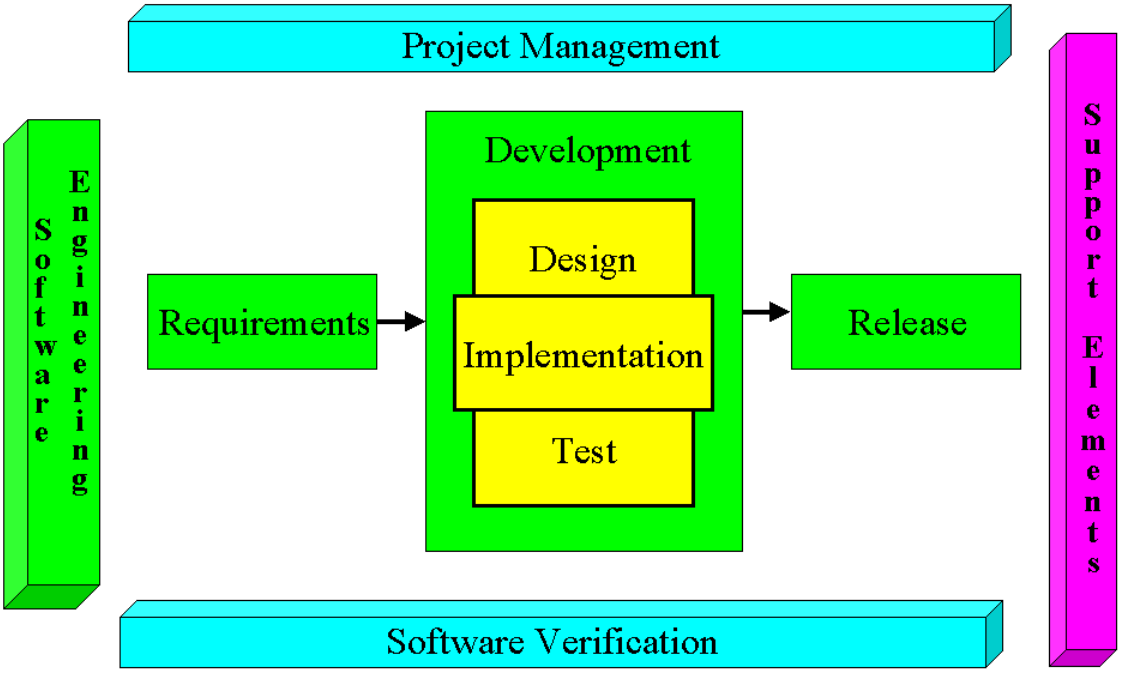

Figure 3. ASCI software program organization.

3.1 Document Organization

Following the program organization, this document has been constructed along the guidelines of the GP&G, tailored to the Sandia site-specific environment. The main areas generally begin with a short introduction, which is followed by a summary table. When needed, additional detail follows the table. After the areas of Software Verification, Software Engineering, Project Management, and Support Elements are described, an assessment tool based on the areas is provided. Appendices list terms used in this document, a mapping from this document to the GP&G, as well as a mapping to QC-1.

Tables in the following sections share common headings—inputs, practices, outputs, and metrics. Phases are significant in Section 3.3 (Software Engineering) as they relate to the code development lifecycle. However, for consistency, the tables that summarize Software Verification, Project Management, and Support Elements also include the same headings, which are described below. The examples provided in the discussion of Inputs and Outputs pertain primarily to the Software Engineering phases.

Inputs

Suggested inputs provide guidance for the artifacts that are needed to complete the practices or to create suggested outputs for the given area or phase. There are two types of suggested inputs: 1) outputs from the previous phase and 2) an artifact that is outside or external to the lifecycle. For example, in the Design Subphase of the Development Phase, one of the suggested inputs is "Outputs from the Requirements Phase." The reader should examine the outputs from the Requirements Phase to determine the inputs to the Design Subphase.

Practices

Each area of the ASCI software program organization (see Figure 3) and each phase of the code development process (Software Engineering) consists of practices that must be accomplished in order to complete the given area or phase. These practices are reflected in the Assessment Checklist (Section 4 and Appendix C).

Outputs

Outputs provide guidance for the artifacts that are required to complete the practices for the given area or phase. There are certain suggested outputs at each phase that will be generated by the various practices: 1) feedback, 2) artifacts that are to be configuration controlled, and 3) issues that are created during the lifecycle.

Metrics

The GP&G defines metrics as "… the activity of collecting information for the characterization, understanding, and evaluation of processes and products." The GP&G states that "only metrics that can be demonstrated to assist in meeting project and/or the V&V program's goals should be chosen." In alignment with the intent that the design, collection, and analysis of metrics contribute to project success and productivity, an authoritative source in Software Metrics, Kan (1997), states:

> Metrics and measurements must progress and mature with the development process of the organization. If the development process is still in the initial stage of the maturity spectrum, a heavy focus on metrics may be counter productive. … In general, the starting metrics ought to be closely related to the final product deliverable.

It is strongly recommended that those who are subject matter experts in the final product be involved in specifying metrics designed to increase product quality and process productivity. Strong customer involvement is also recommended. Metrics are provided for each of the software lifecycle development phases. These metrics can be tailored based on the requirements of the final product.

Named reviews, for which metrics are collected, need to be included in each phase/subphase. These metrics provide the required evidence that the review occurred. Suggested beginning metrics include the following:

- type of review
- date of review
- who performed review
- artifact(s) reviewed
- number of person hours spent
- number of problems/issues found
- number of problems/issues not resolved .

Issues are an expected output of all phases of the development lifecycle. Statistics metrics should be reviewed by project leaders to determine status and to target areas for improvement. Suggested metrics include the following:

- issue ID
- issue submitter
- issue date
- issue severity
- number of issues
- number of open issues
- number of closed issues
- number of deferred issues
- average time issue is open .

3.2 Software Verification

Software verification is achieved through the practices of reviews and testing throughout the software lifecycle. The activities of testing and review ensure that evidence is produced which demonstrates that verification is occurring as needed. Training, education, and experience enable staff to have the ability to carry out necessary software verification practices. Training is described in Section 3.5.4.

Table 3. Software Verification Summary

SOFTWARE VERIFICATION
Overview:
The purpose of Software Verification is to ensure that the released software product complies with software requirements.
Inputs:
• Software requirements • Existing code • Existing lifecycle artifacts
Practices:
♦ Testing: general, unit/integration, regression, verification, installation. ♦ Reviews: of artifacts, including algorithms, numerical methods, requirements trace, design, test plans. ♦ Produce lifecycle artifacts that demonstrate transformation of requirements into product.
Outputs:
• Verified software product • Lifecycle artifacts that demonstrate transformation of requirements into product • Evidence of review and approval of verification-related artifacts
Metrics:
• Code Coverage • Review statistics

Testing

Testing is a critical component of software verification. The goals of testing are 1) to identify errors that need to be corrected and 2) to contribute to user confidence in the code. There are several categories of testing methods:

- general
- unit/integration
- regression
- system software verification
- installation

These tests range from focusing on the internal structural correctness of the software (white box) to the demonstration of high-level requirements that the software is to satisfy (black box).

Specific requirements for testing are provided in section 3.3.2.3. A general discussion of the test categories is provided below.

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

Unit/integration testing covers low-level structural testing of modules and integrated modules prior to full software product testing.

Regression testing can consist of a combination of white box and black box tests and is required after a change has been made to previously tested code. The focus is typically on adequate coverage of the code, ensuring defects are not introduced by the changes and that the changes function properly.

System software verification testing is conducted to demonstrate that specific modeling capabilities function properly without the use of experimental or real data for comparison of results. Tests include analytic solutions, semi-analytic solutions, and idealized solutions. The manufactured solution testing approach may be used to demonstrate specific algorithm implementations.

Installation testing is conducted to confirm that the software installation on the target platform occurred correctly. Installation tests are typically delivered with the software for execution by the end user. These tests may form the basis of customer acceptance tests.

Successful testing of an application code is dependent on the knowledge and expertise of those designing test cases, the knowledge and expertise of those who review test case design, and the results of test execution.

Reviews

Reviews are an important aspect of software verification. Reviews are defined for each lifecycle phase and are divided into three types: technical, quality, and management. The three types of reviews provide verification evidence that technical, quality, and management commitment requirements have been met.

Each phase of the code development process requires one or more reviews. Reviewers may be external or internal to the application team, depending on the type and purpose of the review. Evidence from a review is required, including such attributes as the date, review type, and review results (e.g., defects found, effort expended, issues identified, actions, responsibilities, target dates for resolution of actions). Code development teams are responsible for generating and submitting review evidence and any associated document artifacts.

Produce Lifecycle Artifacts

During the course of following this procedure, the production of artifacts, or objective evidence, is needed. Artifacts provide documentation that is useful in further development of the code, verification of technical soundness, and code maintenance. The guidelines for producing a particular artifact are given in the phase associated with the production of the artifact. Artifacts may be separate entities or combined into single documents as needed. For example, the documentation of requirements and the test plan could be placed in a single document. All artifacts are subject to review (technical, quality, and management). Review evidence is a type of artifact.

3.3 Software Engineering

There are three main phases in the Software Engineering Development Lifecycle: Requirements, Development, and Release (see previous Figure 3). The Development Phase also includes three subphases—Design, Implementation, and Test. Subsequent discussions in this document may use the term *phase* to mean either phase or subphase because both include common areas: inputs, practices, outputs, and metrics.

Each phase includes a summary table followed by additional detail on what the suggested practices actually involve. The practices are listed again in the assessment tool in Section 4 where a recommended AQMC number follows. This number corresponds to the organizational goal level currently recommended by the AQMC as applied to Class A code teams:

3 = should be fully implementing the practice
2 = should be partially implementing the practice
1 = should be planning to implement the practice

Feedback is an important part of the iterative lifecycle. Feedback occurs when the application team discovers that the current phase impacts a previous phase and the impact must be addressed before the current phase can be completed or the next phase addressed. Feedback may result in revisiting a previous phase, through multiple iterations, to rework or reissue a particular deliverable.

The code development process, shown previously in figure 3, consists of phases whose practices and artifacts embody the software application being developed at a point in time. The phases are concerned with actually *doing* the work of building a software application and not specifically concerned with *managing* the work. The practices that contribute to these phases are the core practices of Software Engineering. The execution of a phase may cause portions of a previous phase(s) to be modified. In that case, changes to previous outputs/artifacts shall be modified and verified to the same level of rigor as the original.

The Software Engineering phases are discussed next.

3.3.1 Requirements Phase

Table 4. Requirements Phase Summary

REQUIREMENTS PHASE
Overview:
The purpose of the Requirements Phase is to develop, capture, baseline, and communicate the software product requirements. These requirements are restated, refined, or derived from the system requirements, e.g., requirements from stockpile drivers.
Inputs:
• Requirements: e.g., customer, quality, functional, product, stockpile driver • Expert computational physics and mechanics knowledge, e.g., theory manual, published papers • Numerical algorithm solvers • Issues
Practices:
♦ Derive software requirements. ♦ Document software requirements. ♦ Assess feasibility, if applicable, and generate estimates for budget, resources, etc. ♦ Establish acceptance criteria based on requirements.* ♦ Determine necessary links to other layers of requirements, code, and tests. ♦ Ensure requirements traceability throughout the subsequent software phases. ♦ Review and approve requirement artifacts.
Outputs:
• Requirements (suitable for translation into design and implementation) that have been derived, documented, reviewed, and approved • Traceability links • Evidence of reviews • Configuration-controlled artifacts • Issues
Metrics:
• Issues statistics • Requirements change statistics, e.g., number of requirements (at any given time period), number (or %) requirements changed (added, deleted, modified) over specified time period

* acceptance criteria based on testing methodologies selected; will be described in test plan.

Derive Software Requirements

The Requirements Phase of the lifecycle begins with the input of requirements from any of several sources. These inputs may start out as a stockpile driver, 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. It is then the task of the software

project team to take these inputs, analyze, understand, and derive the software requirements that will be used as the basis for designing and coding the resultant software application.

Document Software Requirements
As the software requirements are derived, they must be documented. Documenting requirements may be accomplished by capturing them in a word processing document, a spreadsheet, or in a more sophisticated tool. Capturing the derived requirements facilitates the prioritization of the requirements. It also leads to developing a specification of how the requirements will be implemented.

Assess Feasibility
The documented, derived requirements are then assessed for their feasibility of being implemented in the next, or upcoming, release of the software application. Whether they will be implemented depends on numerous factors, particularly the perceived priority by the customer or sponsoring organization, the staffing and schedule demands available, and the dependence or effect each requirement has on other parts of the software system. In some cases, assessing the feasibility will result in contacting the originator of the requirement for further clarification or more information, reanalysis of the requirement, or reprioritization of how and when the requirement will be implemented.

Establish Acceptance Criteria
Once requirements are accepted for inclusion into the next release of the application code, it is important to begin the process of establishing acceptance criteria for verifying that the implementation of a given requirement complies with and satisfies the specification of the requirement. Thus, once the application has been prototyped or more formally developed, the acceptance criteria outlined in this Requirements Phase will be incorporated into the test plan completed in the Test Subphase.

Determine Necessary Links/Ensure Requirements Traceability
An important aspect of the Requirements Phase is establishing and maintaining a traceability between a derived requirement and its source or origin. In many cases, requirements for ASCI software applications may extend back through several layers or sources. In general, the traceability between layers requires that for any *what* requirement in a particular layer, there must be some *why* requirements in the previous layer and some *how* requirements in the subsequent layer, assuming that these layers exist. For instance, from an ASCI software project's viewpoint, this means that for any *what* requirement for the software, there must be some *why* requirements in the modeling/simulation and some *how* requirements in the project's application design. As requirements are added or changed, it is important to maintain traceability so that requirements sources are known.

Review and Approve Requirement Artifacts
Finally, before moving into the Development Phase, it is important to ensure that the requirement artifacts (e.g., documented requirements, requirements specification, traceability matrix, acceptance criteria) have been adequately reviewed and approved at the appropriate peer and/or management level. The approved requirements should be base-lined and placed under configuration control so that the design and implementation teams can develop a firm development plan.

3.3.2 Development Phase

Table 5. Development Phase Summary

DEVELOPMENT PHASE
Overview:
The purpose of the Development Phase is to take the output from the Requirements Phase and iteratively perform Design, Implementation, and Test Subphase practices that result in outputs and exit criteria that are sufficient for moving the application code into the Release Phase of the software development lifecycle.
Inputs:
• Expert scientific software development knowledge • Outputs from Requirements Phase • Existing codes, including third party software that may be internal or external to the application code team
Subphases:
DESIGN
IMPLEMENTATION
TEST
Outputs:
• Design Subphase outputs • Implementation Subphase outputs • Test Subphase outputs • Test cases and results • Evidence of reviews • Feedback • Configuration-controlled artifacts • Issues
Metrics:
• See subphases of the Development Phase

The three subphases of Development take place somewhat iteratively without a strict order to the practices involved. For instance, prototyping activities to establish the feasibility of a design concept may commence before the entire design is complete or documented. Unit testing may be designed into the prototype and test results presented to the design team so that the design can be refined prior to formal implementation. The following subphases (Design, Implementation, and Test) illustrate the typical inputs, practices, and outputs that can be expected to occur in this all-important Development Phase.

3.3.2.1 Design Subphase

Table 6. Design Subphase Summary

DEVELOPMENT PHASE
DESIGN SUBPHASE
Overview:
The purpose of the Design Subphase is to describe components in a manner that can be implemented in software. Examples include control flow, embodied mathematical models, data structures, class definitions, and prescribed ranges for data inputs and outputs.
Inputs:
• Outputs from Requirements Phase • Existing codes, including third party software
Practices:
♦ Derive the design. ♦ Communicate the design to the team. ♦ Document the design. ♦ Evaluate impact to requirements (may generate issues). ♦ Plan for testing: initiate development of test plan. ♦ Review and approve design artifacts.
Outputs:
• Derived, documented, reviewed, and approved design document • Test plan (draft) • Evidence of reviews • Feedback generated from Design Subphase • Configuration-controlled artifacts • Issues
Metrics:
• Issues statistics

The Development Phase begins once requirements have been satisfactorily derived, documented, reviewed, and approved. At this point the project team will begin the all-important practices associated with designing the aspects of the software system. These design aspects include such activities as determining the structure of the software system (its design entities and dependencies) and designing the content of the system inputs and outputs and the user and system interface(s). The team will also want to consider any necessary security controls, data structures, new or additional numerical algorithms, and system architecture issues. One or more team members may initiate a prototype of key requirements or functionality that they will bring back to the design team to factor in results or numerical estimates before the design is complete. Another important activity in the Design Subphase is to begin planning for various testing activities that will be required to ultimately verify that requirements have been correctly implemented.

Derive the Design
With requirements from the previous phase in hand or refactored from a previous phase, the development team will work on identifying and specifying the various components and subsystems of the proposed application. The design may take the form of notes from engineering notebooks prepared by various members of the team working independently or it may derive from project meetings where ideas are shared, discussed, and analyzed. The practice of deriving the design will likely be an iterative process based on many discussions and prototypes of various aspects that come out of these discussions.

Communicate the Design to the Team
At some point before moving into a full-fledged implementation subphase, all members of the project team need to be made aware of the design. The project lead or the individual who has been responsible for gathering design notes, reviews, and other design artifacts will be responsible for communicating the design to the entire team. This communication may take the form of a published report, a presentation of design notes, or some combination thereof. Project team design reviews that include customer or sponsor representatives should also be communicated to affected members of the design team.

Document the Design
In communicating the design, some form of documentation is usually produced. However, as the development process matures, the design should necessarily be turned into a document that can be reviewed and approved and included as a product artifact. The design document should be configuration controlled.

Evaluate Impact to Requirements
As the design is derived and communicated, some issues may arise that need to be refactored into the previously identified requirements. These may be feasibility issues related to practicality or resources necessary for accomplishing the implementation of the desired product. Such impacts must be documented and communicated to those involved with project planning and tracking activities.

Plan for Testing
One of the most important aspects of the Design Subphase is to initiate the development of a test plan(s) that will be used throughout the remaining phases of development. Although the completed test plan is not due until the Test Subphase is completed, it is crucial that the design team begin identifying the types of general, integration, regression, software verification, and software validation tests that will be necessary to guarantee the correctness and validity of the application. The nature of the test plan is described in more detail in Section 3.3.2.3, which also includes a discussion of various types of tests.

Review and Approve Design Artifacts
Finally, before moving into the Implementation Subphase, it is important to ensure that the design artifacts (e.g., documented design, draft test plan) have been adequately reviewed and approved at the appropriate peer and/or management level. The approved design artifacts should be base-lined and placed under configuration control so that the implementation and test teams can inherit a well thought-out and documented design plan.

3.3.2.2 Implementation Subphase

Table 7. Implementation Subphase Summary

DEVELOPMENT PHASE
IMPLEMENTATION SUBPHASE
Overview:
The purpose of the Implementation Subphase is to transform the software design into code.
Inputs:
• Expert scientific software development knowledge • Outputs from Design Subphase • Existing codes including third party software • Equations/numerical model/algorithms • Implementation strategies (i.e. language) • Data strategy and model
Practices:
♦ Evaluate impact of implementation to design and requirements. ♦ Translate design into code and other software product artifacts. ♦ Communicate issues with requirements/design team and developers. ♦ Review and approve implementation artifacts.
Outputs:
• Written, reviewed and approved code source and/or executables • Evidence of reviews • Feedback • Configuration-controlled artifacts • Issues
Metrics:
• Issues statistics

Evaluate Impact of Implementation to Design and Requirements
As the implementation proceeds from the simple to the complex, the team will continually evaluate the impact of the implementation to the design. The team will meet frequently to discuss restructuring and integration issues. When necessary, the design will be modified or the requirements will be renegotiated with the stockpile drivers; requirements tracing is extremely important to ensure this.

Translate Design into Code and Other Software Product Artifacts
Design, implementation, and testing are overlapping areas in the Sandia ASCI development environment. Implementation may take place concurrently with design. As code team members identify distinct components or modules of the product, they may spend a few days or weeks translating some aspect of that design into code (prototyping a concept) to determine its

implementation feasibility. Once the code team members have achieved some results, they will then present these to the design team for consideration. As the cycle continues, the implementation team will generate other product artifacts in addition to code. In most cases, theory manuals, user documentation, unit test cases and results, interface specifications, and other outgrowths of implementation will be generated.

Communicate Issues with Requirements/Design Team and Developers
Implementation issues will occur that must be communicated to the design team. Occasionally, significant design changes will result and then these changes must be communicated to all developers who are involved in coding and implementing various components of the system.

Review and Approve Implementation Artifacts
As implementation artifacts are developed and completed, they must be reviewed for completeness and correctness. Test case results must be reviewed to determine that acceptance criteria are met. If not, then another iteration of issues and coding will be necessary. As documentation is prepared, it too must be reviewed. The application team must determine an approval process that goes hand-in-hand with testing and review prior to moving the artifacts out of the Development Phase and into the Release Phase where they will be base-lined and prepared for distribution.

3.3.2.3 Test Subphase

Table 8. Test Subphase Summary

DEVELOPMENT PHASE
TEST SUBPHASE
Overview:
The purpose of the Test Subphase is to identify defects in the software product and to demonstrate that the software product meets its software requirements.
Inputs:
• Outputs from Implementation Subphase • Test plan from Design Subphase
Practices:
♦ Finalize test plan. ♦ Execute test cases found in test plan. ♦ Review test case output using acceptance criteria defined in test plan. ♦ Document test case results. ♦ Retest updated software if acceptance criteria are not satisfied. ♦ Review and approve Test Subphase outputs.
Outputs:
• Developed, executed, reviewed, and approved test plan • Developed, executed, reviewed, and approved test results • Evidence of reviews • Feedback • Configuration-controlled artifacts • Issues
Metrics:
• Issues statistics • Code coverage statistics • Defect statistics

Finalize Test Plan

Test plan development is initiated in the Design Subphase, and some testing is carried out in the Implementation Subphase. Each test plan must identify the class of the software application based on the guidelines described in Section 1.4. The plan must also identify the types of tests that will be conducted based on the class, 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. Every test that will be conducted in the Test Subphase must be described along with acceptance criteria that will be used in the review of test results. Each test must have a specification that contains information to identify the test, test environment, test procedure, and expected test results with acceptance criteria. The test plan must address basic areas of testing: unit, integration, regression, system software verification, installation, and acceptance. Unit

testing is usually conducted during the implementation subphase, but the unit test plan and its results are required by the end of the Test Subphase. See Testing Requirements (below) for a complete discussion of what the test plan should include relative to each of the testing types.

Execute Test Cases
It is expected that some testing is done in the Implementation Subphase. Such outputs will be carried forward to this subphase. For testing that has been identified in the test plan and not performed up until now, a test subteam is responsible for executing and documenting all such test cases.

Review Test Case Output Using Acceptance Criteria Defined in Test Plan
Results from test cases must be reviewed. In cases where unsatisfactory results are obtained, further analysis may be required and, oftentimes, issues may be submitted that will result in the code being reworked to correct the deficiency or oversight. This practice relies on knowledgeable test reviewers and well-defined acceptance criteria so that objectivity can be applied in determining whether or not the code passes the test case criteria.

Document Test Case Results
The results from all tests cases should be documented and added as artifacts to the project's configuration repository. Such test results will form the basis for subsequent reviews or concerns that may arise regarding verification of the software product.

Retest Updated Software if Acceptance Criteria is not Satisfied
In cases where the software code fails to meet acceptance criteria and must be reworked or sent back to the Design and/or Implementation Subphases, it will need to be retested with subsequent reviews against acceptance criteria. New test results will then be documented and added to the project artifacts.

Review And Approve Test Subphase Outputs
Once the software has been successfully tested according to a prepared test plan and all acceptance criteria satisfied, the product is ready to enter the next phase of the lifecycle, Release. Before this phase is initiated, however, it is very important that someone on the project team review and approve all Test Subphase outputs as many of these will be part of the distribution package.

3.3.2.3.1 Test Requirements

The Test Subphase practices center on completing, conducting, analyzing, reconducting (as necessary), and approving the tests that are appropriate for the size, scope, and maturity of the project. The key to meaningful and successful test cases is highly dependent on the knowledge and expertise of the personnel who design the test cases as well as those individuals who review output from the test cases. In the ASCI software development environment, the testing criteria discussed below should be applied. These testing categories are identified in Section 3.2 Software Verification under Testing. The following discussion adds more specifics to the testing categories introduced in that section.

General Testing

- *80% Code Statement Coverage*
  Evidence must be provided demonstrating that at least 80% of the software source statements have been executed through testing. Applying an automated tool that uses a specified set of tests (such as the regression tests) typically provides this evidence.

- *Memory Testing*
  Memory testing is conducted prior to check-in to the configuration control system. It is a white-box testing methodology used to determine that the program is properly using memory. Memory testing is programming-language dependent; some languages do not support memory testing. Memory validity and usage checks can provide useful information. A memory leak can lead to a program prematurely running out of memory or incorrectly overwriting information.

- *Static Compiler Testing*
  Static testing provided by the compiler (for all applicable platforms) is required prior to check-in to the configuration control system. No compiler errors are allowed. Acceptable compiler warnings should be documented as part of the test plan.

Unit /Integration Testing

Unit module testing is conducted prior to check-in to the configuration control system. It is the process of testing the individual units or modules of a program before they are integrated into the software product. Integration testing involves testing part or all of the system to evaluate the interactions among components. Specifications for the test cases must be provided, acceptance criteria must be established, and the source code must be available.

Regression Testing

Regression testing is conducted prior to check-in to the configuration control system. It is conducted after making a change to software (adding functionality, fixing a bug, etc.) to demonstrate that previously tested functionality has not changed and to determine if the change has impacted other aspects of the code. Regression tests are typically a subset of the test cases used to demonstrate software verification.

System Software Verification Testing

This testing consists of using a method or combination of methods to ensure that required functional features satisfy specified requirements. One or more of the following options, as appropriate, should be included in test plans:

- *Manufactured Solution Testing*
  In the Method of Manufactured Solutions an analytical expression, usually as simple as possible, is substituted in the governing partial differential equations (PDEs) and the resulting terms gathered to form a source term. This source term is then used in the code that represents the numerical implementation of the PDEs. An array of source term storage is needed for every grid block or element in the domain. If this array is not available, the code must be modified accordingly. Having to modify the code being tested would be a drawback to the method. The numerical solution is then compared with the analytical expression. By doing a grid refinement, one can verify the expected order of the numerical method. This comparison and verification helps to determine programming errors and numerical errors [Roache, 1998].

- *Analytical Solution Testing*
  This technique compares the code with an analytical solution of the mathematical equations instantiated in the code. Analytical solutions represent simplified solutions to complex problems. Many approximations are usually required to obtain a formulation that can be solved analytically. However, these approximations do permit the testing of the time-

dependent evolution of physical phenomena, e.g., shocks and discontinuous behavior. If the solution does not exist in the literature, it can be resource intensive to develop. Although the solution is analytical, the solution must be translated into a numerical representation that can introduce coding errors.

Because analytical solutions are "exact," the discretization error of the code can be quantified and studied. However, to obtain an analytical solution, simple geometry, boundary conditions, initial conditions, and material models are required, and hence have limited coverage of the code's capability. Even for relatively simple problems, in many cases few analytical solutions are available for 3D geometry. Analytical solutions are "exact" in that they exactly satisfy the mathematical equations, but the form of the analytical solution is in terms of mathematical functions that must be carefully evaluated to get accurate numerical values. Without careful evaluation, inaccurate numerical values can corrupt the comparison with a code.

- *Code Comparison Testing*
  Agreement between a new code and a widely used code can contribute to confidence in the results. It is not required that the two codes being compared be identical, but that they have functionality in common. The basis for this methodology is the assumption that if two independent codes produce the same result, either both codes are correct, or both codes are incorrect in exactly the same way. If possible, code comparison testing needs to be combined with other testing techniques that address typical mistakes with the methodology. When used in this manner, code comparison can greatly contribute to code verification.

Installation Testing

Installation testing is 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 typically 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 Phase, although the installation tests can be designed, reviewed, and approved during the Test Subphase. 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

3.3.3 Release Phase

The Release Phase of the software engineering lifecycle covers practices and activities that must be addressed when a product release is eligible for distribution and support. These activities commence when a new software release is envisioned or when a new version of the release is requested.

Table 9. Release Phase Summary

RELEASE PHASE
Overview:
The purpose of the Release Phase is to manage a production version of the software product that is distributed to customers.
Inputs:
• Outputs from Test Subphase • Request for release • Release distribution process (defined at organization level and tailored by each application software team)
Practices:
♦ Receive and evaluate release request. ♦ Plan and develop release. ♦ Review and approve release. ♦ Create and distribute release. ♦ Support release, as agreed with customer.
Outputs:

• Software product includes code and other designated artifacts
• Operational documentation (may include)
  • Release contents:
  ❑ User documentation, training material, theory manuals
  ❑ Service-level (maintenance) agreement
  ❑ Test cases
  ❑ Installation procedures
• Feedback
• Evidence of reviews
• Configuration-controlled artifacts
• Issues

Metrics:
• Release statistics (types of releases: primary, patch, major, minor, etc.) • Issues statistics

Releases may be preplanned, where the features are identified in the Requirements Phase, carried through the Development Phase, and the release is planned for and scheduled as part of the overall product strategy. On the other hand, once an application is in production, it will be refined, fixed, and enhanced. In this situation, new versions of the product will become eligible

for release and distribution. Depending on the situation, a release may take on all elements of the product or it may include only a subset of the product elements and components. In any case, there are several practices that must be considered and applied as the software product moves from its development environment to the supported production environment. Project teams should tailor and follow a release and distribution management process that is based upon an organizational standard. Such a process should address elements described in the practice descriptions that follow.

Receive and Evaluate Release Request
A product release request may be submitted to the project team as a natural by-product of the Development and Design Phases. In this case, the request will include information that specifies the version, features, platforms and operating systems, and a target release date that coincides with the completion of the Implementation and Test Subphases. The request will also likely include a list of customers or institutions that have been identified to receive the distribution of the product release. A product release request may also be submitted by a new customer who wishes to receive a distribution of an existing or planned release.

Each project team must have some method of receiving and evaluating each release request. The process will include determining what gets released and when; what elements and/or components of the product will be part of the resulting release distribution; how a distribution of the product will be tested and certified for release; and finally, who will be responsible for interfacing with the customer(s) and handling issues that may be submitted against the released product.

Plan and Develop Release
Once the release request has been evaluated and a determination made to proceed with the request, the project team is responsible for planning the activities that must occur prior to base-lining the necessary code and other artifacts that will be distributed. This practice will include planning exactly what will go into the release, what resources are needed to accomplish the distribution of the release, what the schedule will be for accomplishing the release, and what other milestones should be identified for accomplishing the release. Such milestones can include additional installation testing, user documentation, installation instructions, or suggested reviews that should occur. Planning the release may take place early on in the lifecycle, but details and modifications to the original plan are completed in this Release Phase.

Review and Approve Release
When the project team has finished all development activities and created all artifacts necessary for the release, the team will create a baseline that will be moved into a staging area in preparation for distribution. Further code development is deferred to the next scheduled (or nonscheduled) release at this point. Once base-lined, a product undergoes the final steps before being distributed and supported.

Create and Distribute Release
Once approved for release, a software product is eligible for distribution. At this point, the release will be created in an appropriate medium. All included artifacts in the distribution baseline will be identified, and the release will be electronically distributed or packaged for distribution and shipped to authorized customers or requestors. Requestors may be internal customers in the same location and on the same network or they may be external customers, located at remote sites, for whom specialized distribution techniques have been identified.

Each distributed release will contain detailed release notes that provide an overall description of the product and a running history of other releases associated with the project. In addition to code that identifies the application, the distributed release package may include operational documentation in the form of user documentation, training material, and theory manuals. The package will also likely include installation procedure notes and test cases that the customer can optionally run and compare to; in most instances, a service-level agreement will also be part of the package.

Support Release

The service-level agreement specifies 1) the period of time of support and 2) the responsible party in the event of a malfunction or if questions arise on any aspect of the release. This agreement also identifies a point of contact and explains how to submit trouble tickets or issues that may need to be filed against the application code. The project team will track who has requested and received releases and what version of the code each customer has received and installed. That way, in case a release needs to be withdrawn at some point in time, the project team knows exactly who should be contacted and advised.

3.4 Project Management

Project management occurs throughout the entire software development lifecycle. The practices of project management are intended to ensure that adequate funding and resources are available to allow successful completion of deliverables and required software practices. Monitoring of projects provides early warning signs of cost or performance issues that need to be addressed if project milestones are to be completed successfully. The involvement of management in the ASCI software quality program is implemented via the AQMC (ASCI Quality Management Council).

Table 10. Project Management Summary

PROJECT MANAGEMENT
Overview:
The purpose of Project Management is to ensure that adequate funding and resources are available to allow successful completion of deliverables and required software practices.
Inputs:
• Implementation Plans (IPs) • Baseline Change Proposals (BCPs)
Practices:
Project Planning ♦ Submit IP addressing project tasks annually. **Tracking and Oversight** ♦ Review milestone status quarterly. ♦ Issue BCPs, if needed. ♦ Prepare performance reporting on a quarterly basis. **Risk Management** ♦ Incorporate risk identification and risk mitigation into project execution using the BCP.
Outputs:
• Updated IPs • Updated BCPs
Metrics:
• Cost variance by month • Schedule variance by quarter • Completion of milestones and mileposts

3.4.1 Project Planning

Project planning includes preparing a plan that describes how the project will be performed and managed. It typically includes a statement of work, constraints and goals, project deliverables, a project timeline, an assessment of resources that will be needed, and the availability of identified resources.

Submit IP Addressing Project Tasks Annually
Project planning begins with the ASCI Program Plan, which is updated periodically by NNSA with input from the Tri-labs. Sandia then develops implementation plans (IPs) that are written annually. An IP describes individual projects and identified milestones and related tasks, an associated schedule, funding, issues, constraints, and assumptions. In formulating the IP, the principal investigator (PI) identifies the work to be performed and prepares a cost estimate, based on available resources, funding, and his/her experience in projecting such estimates. Implementation Plans are approved by DOE-HQ (DP-10).

3.4.2 Tracking and Oversight

Tracking and oversight involves the tracking and reviewing of projected accomplishments and results with respect to how they are described in the project plan. It also implies taking corrective action as necessary based upon actual accomplishments and results. To that end, selected contents of the IPs are documented and maintained in a Web-based system that

- archives the Work Breakdown Structure (WBS)
- automates data collection for reporting purposes
- provides reporting capabilities
- issues monthly budget updates regarding cost expenditures to PIs

Review Milestone Status Quarterly
Milestones are reviewed and modified on a quarterly basis via a Web-based system that identifies the milestones and their associated due dates.

Issue Baseline Change Proposals (BCPs), If Needed
Whenever changes to the project scope, cost, or schedule are anticipated, the PI, using the Web-based system, must submit a Baseline Change Proposal (BCP). The BCP includes a change description, scope impact, schedule impact, cost impact, justification for the change, and impact of nonapproval. The ASCI Applications program element lead as well as the line manager responsible for the execution of the work must approve the BCP.

Prepare Performance Reports on a Quarterly Basis
For every WBS element, there is at least one milestone that has been identified. Performance reports are prepared on a quarterly basis via a Web-based system that describes the work performed during the quarter relative to meeting the milestone(s). In addition to the brief description of the work performed, the PI also can include supporting documents that were prepared during the quarter. Performance reporting is not, however, limited to this system. Project managers perform informal reviews during the year, which can include one-on-one sessions with the PIs or review sessions in group settings. In addition to these performance reports/sessions, a limited number of external reviews (one or two per year) are conducted on a major milepost and/or major milestone. This forum also provides an opportunity to assign status to the work that has been performed.

3.4.3 Risk Management

Risk management involves identifying, addressing, and mitigating sources of risk before they become threats to the successful completion of a project.

Incorporate Risk Identification and Risk Mitigation into Project Execution
Risk management is incorporated into the IP and the quarterly report. Risks are identified and described in the “Issues and Concerns” section of the quarterly report. Any item that is identified

in this section of the quarterly report is flagged for further review by management to determine the impact on milestone completion. Coupled with the quarterly reporting Web-site is a BCP. The BCP allows mitigating actions to be taken before risks become a threat to successful completion of a project.

3.5 Support Elements

Table 11. Support Elements Summary

SUPPORT ELEMENTS
Overview:
The purpose of Support Elements is to help monitor and correct project plans against performance, conduct reviews of artifact content, train software developers, and document and preserve the results of the project.
Inputs:
• Software requirements • Project planning artifacts • Code artifacts, including those relevant to third party software
Practices:
Requirements Management ♦ Conduct requirements tracing. ♦ Determine requirements ownership and status tracking. **Configuration Management** ♦ Conduct issue tracking of software product artifacts, including requirements. ♦ Perform version control of software product artifacts, including requirements. ♦ Perform release and distribution management. ♦ Engage in ASCI records management. **Third Party Software** ♦ Accept third party software and libraries into the application code domain. ♦ Install, integrate, and control the accepted third party software. **Training (need-based)** ♦ Evaluate training needs on activities necessary for producing software artifacts, use of software tools, needs for understanding of software processes, needs for software verification process and techniques. ♦ Train appropriate project members in use of project management and project tracking and oversight processes. ♦ Train staff on activities necessary for producing software artifacts. ♦ Train staff on how to use software tools. ♦ Train staff on software processes and their implementation. ♦ Train staff on software verification process and techniques.
Outputs:
• Configuration-controlled artifacts • Issues
Metrics:
• Issues statistics

Support elements of the software development lifecycle include requirements management, configuration management, third party software management, and training. These practices are

intended for *managing* the work of building a software system. They help monitor and correct project plans against performance, conduct review of artifact content, train software engineers, and document and preserve the results of the project, which are its artifacts.

3.5.1 Requirements Management

Requirements engineering consists of two significant areas:

- requirements gathering and derivation, which is part of the software engineering lifecycle (Section 3.3.1).
- requirements management. This document treats the requirements management practices as Support Elements.

Requirements management includes practices for requirements tracing, requirements ownership and status tracking, requirements version control, and requirements change control. Version control and change control of requirements are treated as configuration management of requirements (discussed in Section 3.5.2).

Conduct Requirements Tracing
Requirements tracing is keeping track of the original driver for a particular requirement, as well as the corresponding specifications, design issues, and implementation artifacts that reflect that requirement. Tracing is important because when a change to a particular requirement is effected, it is essential that the change be applied against all other product artifacts that reflect any part or all of the requirement.

Determine Requirements Ownership and Status Tracking
Requirements ownership and status tracking imply a knowledge of where a particular requirement originated, who or what component is responsible for implementing it, and who is responsible for managing any associated changes against that requirement over the lifetime of the software product. As individuals come and go from the project and as modules are added or deleted or rearranged, it is extremely important to ensure that requirements are not overlooked or abandoned. It is also very important to know where and when the requirement was implemented in the code and how it was verified.

The process for managing requirements is critical to ensure that ASCI codes share a common understanding from the various viewpoints at any point in time. Requirements management will also ensure that projects are managed to customer requirements.

3.5.2 Configuration Management

Configuration management includes identifying the configuration items in a system, controlling the change and release of those items throughout the lifecycle, recording and reporting the status of the items and associated changes, and managing the completeness and traceability of the items. In short, a configuration management system should provide a stable environment for iterative development and production activities. Required configuration management practices for controlling and managing software artifacts are

- issue tracking
- version control
- release and distribution management
- records management

Conduct Issue Tracking of Software Product Artifacts
Issue tracking is the process of recording and tracking all changes that occur to any product artifacts throughout their lifetime. Issue tracking allows the submittal of enhancement requests, problem and defect reports, and inquiries. Most issue tracking systems provide a capability of tying the requested change to a particular code module (or modules) and controlling who can work on the change request at particular status points in a module's existence.

Perform Version Control of Software Product Artifacts
Version control of software product artifacts implies the availability of a controlled, shared project repository (library) where artifacts are stored and accessed. Each project needs to follow a documented process describing how to identify project artifacts that will be kept in the repository, how to access and version those artifacts, how to identify when product baselines will be created and how they can be changed and by whom, and when software is ready to be released and distributed to internal or external customers.

Perform Release and Distribution Management
Release and distribution management involves determining what will go into a release, when it is ready to be distributed (and to whom), and how a given release will be supported and tracked throughout its lifetime. Section 3.3.3 describes the entire release and distribution management practices in more detail. Configuration management is used to control how project artifacts will be base-lined and preserved, to identify to whom and when releases are distributed, and to be able to recreate or distribute a given release.

Engage in ASCI Records Management
Records management is a corporate requirement. It involves the planning, organizing, training, and other managerial activities related to the creation, maintenance, use, and disposition of records. The Sandia ASCI Records Management Program strives to meet its records management needs by fostering an understanding of the importance of recorded information generated or received by Sandia. This program also strives to teach Sandians their responsibility in the creation, use, maintenance, and disposition of records; to provide training and support for the implementation of best business practices with regards to Sandia ASCI records; and to incorporate federal requirements into standardized tools for information management at Sandia.

3.5.3 Third Party Software

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

Place Accepted Third Party Software into Application Code Domain
Third party software might serve as an input into several of the Software Engineering phases described in Section 3.3. Sandia manages the ongoing development and maintenance of third party software once it enters the application code domain. These third party software packages are required to pass a quality assurance procedure and then are configuration-controlled. If third party software is modified by the ASCI application team, then either the team assumes primary responsibility for these changes (in which case it is no longer third party), or such changes are coordinated with the third-party supplying organization for inclusion in future updates and releases. Third party software must be evaluated on a case by case basis to determine its appropriate class (Table 2, Section 1.4)

Install, Integrate, and Control the Accepted Third Party Software
Besides furnishing artifacts that verify the integrity of the supplied third party code, the supplying organization is expected to include instructions, code, test cases, and user information that allows the Sandia developer to successfully install, integrate, and appropriately control the code. Each code team should have a plan in place that describes the criteria for accepting third party software into its domain. Such software and its associated artifacts, once accepted, should be managed according to a common software configuration management process.

3.5.4 Training

Training addresses the importance of the "human asset" in the ASCI application code development process. The staff involved in the practices of this document must be highly trained and educated in scientific software development, algorithms, and/or computer science. Specific project and tool training related to software development, software verification, and project management will be planned and tailored in an individualized, need-based implementation. Training in the following areas will be conducted as project needs dictate:

a. project management and project tracking and oversight processes
b. activities necessary for producing software artifacts
c. use of software tools
d. software processes and their implementation
e. software verification process and techniques

As training needs evolve, the code teams will follow a graded approach in determining the specific types of training classes or opportunities that are needed for their environments. For example, self-directed learning exercises using Web-based tools can be a method for providing training. Vendors offer extensive classes in the use of support tools. Many classes are offered by Corporate Training in classroom format or video downloads, covering current software engineering practices.

4 Assessment Tool & Gap Analysis

This section includes an assessment tool based on the practices and suggested outputs of this document. Periodically, the AQMC will review this assessment tool and modify it as necessary. The assessment tool provided by the AQMC will list practices and the current organizational goal level for each of the practices. The assessment tool is a process improvement mechanism that is used to

- set measurable goals for software engineering practices and outputs
- evaluate the current state of software engineering practices
- compare the current state of software engineering practices to a desired state (perform a gap analysis)
- gather information on an application code team's interpretation of compliance
- compile an overall consistent organizational evaluation of software engineering practices

The results of the assessment should aid management in resource allocation, risk identification, and priority identification.

The assessment tool organizes practices as they are introduced and discussed in Section 3. Software Verification is not included as a stand-alone category in the tool because the primary components of software verification, reviews and testing, are folded into various practices under the phases of Software Engineering. Support Elements are addressed by practices of requirements management, configuration management, and various training activities pertaining to lifecycle support.

The assessment tool will be deployed with the following strategy:

- The AQMC will initially set the values in the tool based on the consensus of the council members.
- Code teams will do a self-assessment and gap analysis, which establishes implementation priorities for the individual teams.
- An independent assessment of the code teams will follow the self-assessments.
- The AQMC will revise the values in the tool approximately one year after the assessment report is accepted.

The assessment tool includes a column for evaluation of the application code team's practices by an independent assessment team. This team will be appointed, as needed, by the AQMC for calibration of evaluation results at the Sandia ASCI-organizational level. The AQMC will direct application code teams to use this tool periodically to compare their current practices to the Sandia ASCI Code Development Practices. This 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 application code teams can observe how they are improving over time by comparing previous assessments to current assessments.

The assessment tool will provide the AQMC with a mechanism for identifying best practices that can be communicated and leveraged among application code teams. For instance, if a software development team chooses to prototype a practice, the team can do so without adding it to the assessment tool.

Note: This tool is designed to identify current status and provide management with information to allocate resources and is not intended as a goodness evaluation, certification, or verification exercise.

Instructions for Completing Assessment Checklist

The details of the activities that compose each practice are not listed separately in the Assessment Checklist. 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. However, if the AQMC recommendation for a particular practice, such as "Finalize test plan," is 3, 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 3 in its self-assessment.

Definitions of the columns in the Assessment Checklist are provided below. Following the definitions is an example of an Assessment Checklist that has been filled in for demonstration purposes only. A blank checklist is provided in Appendix C.

(1) Application Name/Class/Assessment Date

This column includes the name of the ASCI application code, the designated class of the code, and the date of the assessment.

(2) AQMC Requires

The values in this column are determined by the AQMC and modified as determined by the council. A value of 3 indicates that the council requires that application code teams follow this practice by fully implementing it.

In general, the council will raise the bar (higher value) for a particular practice when it reaches consensus with application code teams that the practice adds value to the process and is cost effective. The council will remove the practice or lower the bar (lesser value) for a practice if it deems that the practice is not cost effective and/or it adds little or no value.

(3) (Application) Code Team Evaluation

This is the column the code evaluation team fills in to determine where they are in terms of performing or implementing all recommended practices. A code team will select a value of 0–3 or NA based on the criteria specified below.

3 The application code 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 code 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 be in place, and the team needs to be following this documented process.

2 The application code 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. Additional resources most likely will be needed to raise this practice to fully implemented.

1 The application code team has proposed the implementation of this practice but has little or no evidence yet to support implementation. At this level, it is typical that resources have not yet been identified and allocated for fulfillment of the practice. Activities and resources for this practice are being planned.

0 The application code team has not yet addressed the implementation of this practice.

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

Note: Specific guidelines for selecting assessment values will be provided by the AQMC for each entry in the Assessment Checklist.

(4) Assessment Team Evaluation
As needed, the AQMC will appoint a core assessment team to review the current state of practices performed by each team. The AQMC will use the same scale as the application code team (see (3) above).

(5) Comments/Evidence
This column is intended to record comments about an application code 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 full or partial implementation. Either the application code team or the assessment team may enter information in this column. The author of the comment should be clearly identifiable.

(6) Completed By
This line indicates the person (code 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.

Application code teams should use this tool annually to determine how closely they are adhering to the Sandia ASCI Code Development Practices. In addition to highlighting areas that are appropriate for increased improvement efforts, the application code teams can observe how they are improving by comparing the scores of various practices from one assessment period to the next.

Sample Assessment Checklist for ASCI Apps Software Development Areas

(1) Application Name: **MADRE** Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
Practice	3=Fully 2=Partially 1=Plan to	3=Fully 2=Partially 1=Plan to 0=Not addressed NA - not applicable	3=Fully 2=Partially 1=Plan to 0=Not addressed NA - not applicable	Use this area to explain why NA is selected as a response to columns (3) or (4) and to demonstrate evidence for other responses as needed.
Software Engineering				
1. Requirements Phase				
1a. Derive software requirements. *Section 3.3.1*	3	3	3	Requirements to be implemented in software are derived based on stockpile drivers.
1b. Document software requirements. *Section 3.3.1*	2	3	3	Requirements for released version 1.0 are documented in requirement's document V1.0.
1c. Assess feasibility, if applicable, and generate estimates for budget, resources, etc. *Section 3.3.1*	1	1	1	Team recognizes the value in this practice; however, schedule does not always permit analysis at Requirements Phase. Often defer this until Implementation Subphase.
1d. Establish acceptance criteria based on requirements. *Section 3.3.1*	1	2	2	A process for establishing acceptance criteria exists; criteria is identified, but not all areas of process are being addressed.
1e. Determine necessary links to other layers of requirements, code, and tests. *Section 3.3.1*	1	2	2	This practice is part of the documented MADRE RM process V1.5.
1f. Ensure requirements traceability to other product artifacts throughout subsequent software phases. *Section 3.3.1*	1	2	2	This practice is part of the documented MADRE RM process V1.5.
1g. Review and approve requirements artifacts. *Section 3.3.1*	1	1	1	Evidence of technical, quality, and management reviews does not exist.

(1) Application Name: MADRE Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
2. Development: Design Subphase				
2a. Derive the design. *Section 3.3.2.1*	2	1	1	Not formally done at this time due to schedule constraints.
2b. Communicate the design to the team. *Section 3.3.2.1*	3	3	3	Periodic meetings held; email sent out on regular basis.
2c. Document the design. *Section 3.3.2.1*	1	N/A	1	Small code team, application team does not see value in this. Assessment (A) team – needed to support release of code.
2d. Evaluate impact to requirements. *Section 3.3.2.1*	1	0	0	Not planning to evaluate impact; if management says "do," code team will design and implement. A-team – need to evaluate if design impacts derived requirements (not based on stockpile driver). This practice is needed to keep consistency between requirements and design.
2e. Plan for testing: initiate development of test plan. *Section 3.3.2.1*	1	3	2	Testing is informally discussed.
2f. Review and approve design artifacts. *Section 3.3.2.1*	1	2	2	Reviews are performed when design artifacts are created; however, creation of artifacts sporadic.
3. Development: Implementation Subphase				
3a. Evaluate impact of implementation to design and requirements. *Section 3.3.2.2*	1	1	1	Feedback of issues into previous phases not yet formalized.
3b. Translate design into code and other software product artifacts. *Section 3.3.2.2*	3	3	2	Code is being produced; however, there is little evidence that implementation represents design.
3c. Communicate issues with requirements/design team and developers. *Section 3.3.2.2*	3	3	3	Team communicates via periodic meetings, group email, etc.

(1) Application Name: MADRE Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
3d. Review and approve implementation artifacts. *Section 3.3.2.2*	1	1	1	No evidence that review of implementation artifacts was occurring.
4. Development: Test Subphase				
4a. Finalize test plan. *Section 3.3.2.3*	1	2	2	General, unit, and regression testing are included in test plan. Unit testing not being done at this time. Installation plan is not complete.
4b. Execute test cases found in test plan. *Section 3.3.2.3*	2	2	2	All required test cases not always executed prior to check-in to configuration system.
4c. Review test case output using acceptance criteria defined in test plan. *Section 3.3.2.3*	3	2	2	Not done in all cases.
4d. Document test case results. *Section 3.3.2.3*	1	1	1	Team sees value in test cases being a controlled artifact; however, resources to do this not available.
4e. Retest updated software if acceptance criteria is not satisfied. *Section 3.3.2.3*	2	1	1	Retesting is not consistently carried out.
4f. Review and approve Test Subphase outputs. *Section 3.3.2.3*	1	2	2	Informal reviews occurring.
5. Release Phase				
5a. Receive and evaluate release request. *Section 3.3.3*	2	2	2	Process in place—not consistently followed.
5b. Plan and develop release. *Section 3.3.3*	2	2	2	Process not consistently followed.
5c. Review and approve release. *Section 3.3.3*	3	2	1	Found evidence that products have been released without approval.
5d. Create and distribute release. *Section 3.3.3*	3	2	2	Releases that are successfully created are distributed. Some releases not distributed to all specified customers.

(1) Application Name: MADRE Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
5e. Support release, as agreed with customer. *Section 3.3.3*	1	N/A	N/A	No agreement to support release in place.
Project Management				
6. Project Planning				
6a. Submit IP addressing project tasks annually. *Section 3.4.1*	3	3	3	IP for FY01 submitted and current.
7. Tracking and Oversight				
7a. Review milestone status quarterly. *Section 3.4.2*	3	3	3	FY01 IP reviewed Q1 and Q2.
7b. Issue Baseline Change Proposals (BCPs), if needed. *Section 3.4.2*	3	3	2	Need for BCP exists; however, changes have not been implemented.
7c. Prepare performance reports on a quarterly basis. *Section 3.4.2*	3	3	3	Quarterly performance reports exist and are complete.
8. Risk Management				
8a. Incorporate risk identification and risk mitigation into project execution using the BCP. *Section 3.4.3*	2	2	2	BCP indicates dependence on another project that is 2 months behind schedule.
Support Elements				
9. Requirements Management				
9a. Conduct requirements tracing. *Section 3.5.1*	1	2	2	The Req. Mgmt. Process is written and is ready to be implemented using the DOORS tool.
9b. Determine requirements ownership and status tracking. *Section 3.5.1*	1	2	2	The Req. Mgmt. Process is written and is ready to be implemented using the DOORS tool.
10. Configuration Management				
10a. Conduct issue tracking of software product artifacts, including requirements. *Section 3.5.2*	3	3	2	An issues tracking tool is in place; however, it is not being consistently used by team members to capture issues.

(1) Application Name: MADRE Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
10b. Perform version control of software product artifacts, including requirements. *Section 3.5.2*	3	2	2	Code and user documentation is version controlled but other product artifacts are not stored in repository as defined by code team implementation plan.
10c. Perform release and distribution management. *Section 3.5.2*	3	2	2	The process for this is written but has not yet been completely implemented.
10d. Engage in ASCI records management. *Section 3.5.2*	1	1	1	Still in planning stage.
11. Third Party Software				
11a. Accept third party software and libraries into the application code domain. *Section 3.5.3*	3	2	2	Third party software plan is implemented and followed.
11b. Install, integrate, & control the accepted third party software. *Section 3.5.3*	3	2	2	Third party software plan is implemented and followed.
12. Training				
12a. Train appropriate project members in use of project management and project tracking and oversight processes. *Section 3.5.4*	2	NA	NA	Project management is in full compliance with organization requirements and has necessary skills. Training not needed at this time.
12b. Train staff on activities necessary for producing software artifacts. *Section 3.5.4*	1	1	1	Team recognizes value of this practice; however, funding and resources not available for providing team with tools to produce artifacts consistently or to train members on use of tools.
12c. Train staff on use of software tools. *Section 3.5.4*	2	3	2	Some staff not using issue tracking tool; may be a need for training.
12d. Train staff on software processes and their implementation. *Section 3.5.4*	1	2	2	Plans in place to train team. Insufficient resources to complete.

(1) Application Name: MADRE Application Class: A Assessment Date: September 27, 2001	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
12e. Train staff on software verification process and techniques. *Section 3.5.4*	1	1	2	Team is very cognizant of verification methods for their application. Training not needed at this time.
Total Number of Areas	**12**			
Total Number of Practices	**46**			
(6) Completed By: (print name and date) (signature)	---------- ----------			

References

Since this document is an extension to the *ASCI Program Software Quality Engineering: Goals, Principles, and Guidelines*, the authors are assuming the references cited in the GP&G are also valid for this document. Only when referenced directly is a work denoted in the list that follows.

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

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.

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

Guidance. The following are documents that provide additional information that is useful in developing and implementing Sandia ASCI V&V practices:

Kan, S. H. *Metrics and Models in Software Quality Engineering*. Reading, MA: Addison-Wesley Longman, Inc., 1997.

Myers, Glenford J. *The Art of Software Testing.* New York: John Wiley and Sons, 1979.

Roache, Patrick J. *Verification and Validation in Computational Science and Engineering*. Albuquerque: Hermosa Publishers, 1998.

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.

Pilch, M., T. Trucano, D. Peercy, A. Hodges, E. Young, and J. Moya. *Peer Review Process for the Sandia ASCI V&V Program: Version 1.0,* SAND2000-3099, Albuquerque: Sandia National Laboratories, January 2001.

Harris, R., D. Cuyler, J. Abbot, et al. *SPE Process Definition, Established by the Software Product Engineering Technical Working Group Organization 9500,* Draft. Albuquerque Sandia National Laboratories, March 2001. Available at http://wfsprod01.sandia.gov/, then search by title = “SPE Process”, or use Advanced search and Document ID = WFS003551.

Appendix A: Glossary and Acronyms

Glossary

acceptance criteria The defined value, or range of values (usually quantitative), expected from a test case execution to demonstrate fulfillment of software requirements.

artifact A deliverable or work product that is the output of some phase of the software development lifecycle. A configuration-controlled artifact is an artifact that is stored in a corporate repository (library) and changes to it are controlled via reported issues.

configuration control An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration artifacts.

derived requirements Those code requirements that result from analyzing and refining the software requirements and determining what will actually be coded.

feedback Information from one phase of the software lifecycle that is fed back to one or more previous phases. The purpose of feedback is to provide an iterative loop from one phase or sub-phase to another and to establish a mechanism for continuous improvement.

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

lifecycle development A model for software development that consists of phases and ensures documentation of technical adequacy throughout the lifetime of software from conceptualization through retirement.

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

reviewer An independent person (someone who did not produce the work or item being reviewed) qualified to perform a review.

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

- **management -** An evaluation performed to verify that *commitments* (for the current phase) 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 technical requirements.

software engineering The activities that an organization consistently employs to ensure that it produces correct and consistent software products effectively and efficiently.

software process A set of activities, methods, and practices for developing and maintaining a software product and its associated artifacts.

software process management The activities of monitoring, evaluating, and improving the software process or processes.

software product One or more artifacts, usually including code, given to the customer.

software quality The development and description of software quality policies, goals, metrics, assessment means, and assurance plans.

software quality management The software quality definition activities, followed by the appraisal of current quality practices against the organization's quality assurance plan, plus the development of organizational support for software quality improvement plans.

software requirements The subset of the system requirements specifically designated to be implemented in software.

software verification The process of determining whether the released software product complies with specified requirements (software requirements).

support elements The practices that the organization performs aimed more at *managing* the work of building a software system rather than the actual *building* of the system.

validation The process of evaluating the mathematical formulation to ensure that it adequately describes the problem of interest, i.e., that the computer simulation adequately represents the real world. [Outside the scope of this deployment document.]

verification The process of determining whether or not the mathematical formulation is solved correctly, i.e., whether the computer simulation correctly represents the conceptual model and its solution. When the numerical model forms the basis for the software requirements, verification is equivalent to software validation.

Acronyms

AL	Albuquerque Office (of DOE)
AQMC	ASCI Quality Management Council
ASCI	Accelerated Strategic Computing Initiative
BCP	Baseline Change Proposal
DOE	Department of Energy
DP	Defense Programs
DSW	Directed Stockpile Work
GP&G	*ASCI Software Quality Engineering: Goals, Principles, and Guidelines*
HQ	Headquarters
IP	Implementation Plan
NNSA	National Nuclear Security Agency
PDE	partial differential equation
PI	principal investigator
QC-1	*DOE/AL Quality Criteria (QC-1)*
R&D	research and development
Sandia	Sandia National Laboratories
SQE	software quality engineering
V&V	Verification and Validation
WBS	Work Breakdown Structure

Appendix B: Mapping and Tailoring Methods

The tables in this appendix provide the evidence of compliance of this document with the GP&G. Documents that were consulted for the compilation of the GP&G (Software Standards, Modeling and Simulation Standards, Nuclear Facilities Standards, Customer Expectations Standards, etc.) are not mapped directly from this document, but are mapped from this document through the GP&G. The GP&G is the mechanism that passes along appropriate requirements from these various standards to this deployment document.

Table 12 provides the mapping from the figure on page 4 of the GP&G (column 1) to the corresponding practices in this deployment document (column 2). This table summarizes the site-specific tailoring and grading performed for this deployment document.

Table 12. Mapping of Key Elements to Practices

Goals, Principles, and Guidelines (Figure Pg. 4)	Sandia National Laboratories ASCI Applications Software Quality Engineering Practices
Guidelines	**Activities**
Software Verification	**Mapping / Tailoring Comments**
Unit Testing Regression Testing	Type of white-box testing. Section 3.2 and Section 3.3.2.3. The GP&G defines this as the "activity of regularly building the code…" Section 3.3.2.2, Implementation Subphase. "… and executing a series of tests designed to verify that the code works as expected for all computational platforms supported." Demonstrating that code works as expected or complies with requirements and acceptance criteria is the purpose of software verification. Software verification is achieved through the fulfillment of the lifecycle. Section 3.2.
Analytic Comparisons	Acceptable method for comparing results of test case execution. Section 3.3.2.3.1.
Code Comparisons	Accomplished by code reviews. Section 3.2 and Section 3.3.2.3.1.
User Acceptance Testing	Demonstrating that the application software meets user needs. User needs are captured in the Requirements Phase (Table 4) and carried through subsequent phases (Development and Release). Requirements are tested in Test Subphase.
Training	Software verification training is a component of Training Support. Section 3.5.4.
Software Engineering	**Mapping / Tailoring Comments**
Lifecycle Management	Lifecycle Management is a component of Project Management, Section 3.4, Table 10 and associated practice discussion.
Configuration Management	Section 3.5, Table 11 and associated practices discussion.
Measurement Metrics	Section 3.4, Table 10 and associated practices discussion.
Reviews/Assessments	Section 3.4, Table 10 and associated practices discussion.
Process Improvement	Glossary "software process management" and Section 2, AQMC.
Training	Software engineering training is a component of Training Support, Section 3.5.4.
Project Management	**Mapping / Tailoring Comments**
Risk Management	Risk Management is a component of Project Management. Section 3.4, Table 10 and associated practice discussion in Section 3.4.3.
Requirements Management	Section 3.5, Table 11 and associated practices discussion.
Project Planning	Section 3.4, Table 10 and associated practices discussion.

Goals, Principles, and Guidelines (Figure Pg. 4)	Sandia National Laboratories ASCI Applications Software Quality Engineering Practices
Tracking and Oversight	Section 3.4, Table 10 and associated practices discussion
Process Management	Glossary “software process management' and Section 2, AQMC.
Training	Project Management training is a component of Training Support, Section 3.5.4.

Tables 13, 14, and 15 provide a mapping from the GP&G—for Software Verification, Software Engineering, and Project Management—as represented by Key Elements to this deployment document.

Table 13. Mapping of Deployment Practices to Key Elements of Software Verification

Stating Goals, Identifying Principles, and Selecting Guidelines for SQE (GP&G, Pg. 7)			Sandia National Laboratories ASCI Applications Software Quality Engineering Practices
Guideline Area	**Activities**	**Key Elements**	**Practices**
Software Verification			**Mapping Comments**
	Technical Reviews	Technical Soundness Static Analysis	Technical review, Glossary and Section, 3.2 (under Reviews)
	Unit Testing Regression Testing		White-box testing technique, Section 3.2, and Table 8.
		Traceable, repeatable component test	Section 3.3.2.3, Table 8, Test Subphase outputs. All artifacts identified for configuration control, such as test cases, will be “repeatable.” Traceability is maintained throughout entire lifecycle, including test subphase.
		Building the Code	Section 3.3.2.2, Table 7, Implementation Subphase.
		Executing tests	Section 3.3.2.3, Table 8, Test Subphase.
		Feature-based test suite for multiple platforms	This is an example of a black-box requirement-based test. The purpose of the Test Subphase is to develop and execute test cases that demonstrate that a given software product meets software requirements. This is application dependent; application codes with requirements to run on multiple platforms will have tests associated with this requirement.
	Comparison Techniques	Analytic solutions Other codes results	Acceptable methods for evaluation of test results. Test Subphase Table 8 and associated discussion, Section 3.3.2.3.1.

	User Acceptance Testing	Applicability Evaluation	The GP&G defines User Acceptance. testing as " the activity of determining if the work products satisfy the needs of the intended user's". The demonstration that any type of requirement has been met is an output of the Development Phase and the purpose of Software Verification. Section 3.2, Figure 2 and Table 3. Software requirements include "user" requirements—the requirements the software is to satisfy. These will then be "evaluated" by review of test cases execution.
		Usability Evaluation	Any "usability" requirements will be captured as appropriate in software requirements. These will then be "evaluated" by review of test cases execution. The fulfillment of any type of requirement. Section 3.2, Software Verification.
		Code Confidence Results Credibility	Code confidence and results credibility are goals, which map to principles, then guidelines. This deployment document meets goals by mapping principles to guidelines.
	Training	Verification methods and techniques	Section 3.5.4, Training.

Table 14. Mapping of Deployment Practices to Key Elements of Software Engineering

Stating Goals, Identifying Principles, and Selecting Guidelines for SQE (GP&G Pg. 7)			Sandia National Laboratories ASCI Applications Software Quality Engineering Practices
Guideline Area	**Activities**	**Key Elements**	**Practices**
Software Engineering			**Mapping Comments**
	Life-Cycle Management	Time-based work flow Requirements, design, construction, test, support activities	Glossary, Figure 3 and associated discussion. Section 3.3, Tables 4–9. Figure 3, Section 3.3.1, Requirements Phase, Section 3.3.2.1, Design Subphase; Section 3.3.2.2, Implementation Subphase; Section 3.3.2.3, Test Subphase, and Section 3.5, Support Elements.
	Configuration Management	Version Management Issue Tracking Release Management	Section 3.5.2, Configuration Management and Section 3.3, Software Engineering (introductory discussion). Section 3.3.3, Release Phase
	Measurements and Metrics	Software Products Software Process	Section 4, Assessment Tool.
	Reviews and Assessments	Management Reviews Technical Reviews	Glossary and Section 3.3, each lifecycle phase.
	Process Improvements	Engineering Process Baseline Identified Improvements Improvement Implementation	Section 4, Assessment Tool. Section 3.5.2, Configuration Management. Section 3.5.2, Issue Tracking practice.
	Training	Software practice methods and techniques	Section 3.5.4, Training.

Table 15. Mapping of Deployment Practices to Key Elements of Project Management

Project Management			**Mapping Comments**
	Risk Management	Risk Assessment Risk Control	The GP&G defines risk management as "The activity of identifying, addressing, and mitigating sources of risk before they become threats to successful completion of a project." Section 3.4.3, Risk Management.
	Requirements Management	Gathering, documenting, verifying, managing change to requirements	Section 3.5.1, Requirements Management and Section 3.3.1, Requirements Phase.
	Project Planning	Statement of Work Constraints and Goals Implementation Plan Resource Assessment	Section 3.4.1, Project Planning.

Project Management			Mapping Comments
	Tracking and Oversight Tracking and Oversight	Actual results vs. planned results Corrective Action	Section 3.4.2, Tracking and Oversight. Table 4, Assess feasibility. Section 3.4, Project Management.
	Process Management	Process documentation & plans. Tech. Improvement Improvement leverage	AQMC responsibility, Section 2 and Assessment Tool, Section 4.
	Training	Project management methods and techniques	Section 3.5.4, Training.

Table 16 provides a mapping from the Sandia ASCI software quality program (this deployment document and the GP&G) to QC-1. Although QC-1 was evaluated, and appropriate items passed along to this document via the GP&G, this additional mapping is provided to emphasize the importance of the standard to nuclear weapon work.

Table 16. Mapping of Deployment Practices to DOE/AL's QC-1

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
II. BASIC REQUIREMENTS **1.0 FUNDAMENTALS OF QUALITY MANAGEMENT** Quality is conformance to customer requirements and expectations. Quality is enhanced by manufacturable, robust designs supplemented by a process of continuous improvement which focuses on the prevention of errors and reduction of variability in processes, products, and services. Quality is measured by the use of appropriate metrics to assess its effectiveness in reducing operating costs, increasing productivity, and keeping the total quality cost to a minimum.	Customer requirements for software are defined in the GP&G. See mapping from the GP&G to this deployment document.
2.0 ORGANIZATION The contractor shall establish and maintain a documented quality system as a means of ensuring that product conforms to specified requirements. Management shall issue quality policy and delegate administration and oversight of the quality system to a responsible, independent, and authoritative element of the organization with clear access to top management.	For software, the quality system is defined in the GP&G and this deployment document. Management oversight of the software quality system is the responsibility of the AQMC as described in Section 2.0. The description of the organization management system, as it applies to criteria other than software, is outside the scope of the software deployment document.
3.0 QUALITY MANAGEMENT The quality system shall be documented and maintained, with adequate provisions for internal checks and balances and management involvement. The system shall promote an environment that provides for individual responsibility and accountability for quality. The system shall be capable of objectively evaluating quality effectiveness and implementing needed improvements.	Internal checks of the software quality system via the assessment tool are described in Section 4. Management involvement in the software quality system is described in Section 2. Section 3.5.3, Engage in Record's Management, " … Sandians their responsibility in the creation, use, maintenance, and disposition of records, to provide training and support for the implementation of best business practices with regards to Sandia ASCI records." Quality Management of program other then software is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
3.1 CONTINUOUS IMPROVEMENT PROCESS A quality improvement process which focuses on the prevention of errors and the reduction of variability shall be an integral part of the quality system. This process should be tailored to fit site specific operations.	The lifecycle process (Figure 3 and Section 3) with reviews at each lifecycle phase, establish a methodology to prevent software errors. Continuous process improvement as applied to elements other then software is outside the scope of this deployment document.
3.2 PREVENTION VS. DETECTION The quality system shall focus on the prevention of errors and nonconformance and promote building quality into products and processes. Fundamental methods, such as design of experiments, prototyping, process capability studies, Pareto analyses, and statistical process controls are examples of methods useful to: a. characterize processes; b. continually reduce product and process variability; c. identify and minimize unstable or error-prone processes; and d. provide early feedback of engineering and manufacturing data to determine e. the need for product or process changes.	Statistical process control is outside the scope of this deployment document.
3.3 QUALITY COSTS The cost of nonconformance plus the cost of conformance and/or other appropriate metrics shall be utilized for performance measurement, problem identification, and problem prevention.	The description of the system to address the definition, requirements, and control of Quality costs is outside the scope of this deployment document.
4.0 TRAINING A formal training and education program shall be established for all personnel involved in assembly, production, manufacturing, inspection, test, repair, disassembly and administrative support activities. These personnel shall be reevaluated at intervals not to exceed three years. In addition, personnel performing special processes shall require certification based on written qualification/ certification procedures. Appropriate records of training, qualification, certification and reevaluation shall be maintained.	Training within the scope of software issues is need based, as described in Section 3.5.4. Other training is outside the scope of this deployment document. The description of the record system is outside the scope of the software deployment document. Software training records maintained as part of ASCI records program.
5.0 EARLY INVOLVEMENT The organization responsible for design shall ensure that production and quality requirements are incorporated in the design process as early as feasible. The design process shall provide for the timely identification and evaluation of key	For software, design is subject to three reviews: technical, quality, and management. These reviews ensure that requirements are translated into the design.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
elements that are critical to program success and shall provide an objective means to measure design, product, process maturity, and production readiness.	
6.0 ESTABLISHING AND VALIDATING REQUIREMENTS The following shall be applied to assure that the initiation of research and development activities includes plans to identify customer requirements and methods to meet those requirements.	Management review, at each phase of the lifecycle, ensures that commitments (including customer requirements) have been satisfied. Outputs of any phase, including Requirements Phase, are verified for conformance to established requirements.
6.1 CUSTOMER REQUIREMENTS There shall be a process for identifying both internal and external customers and documenting their requirements, including changes to requirements, and or verifying that process outputs meet the established requirements.	Section 3.3.1 requires that all requirements be identified, including customer requirements (external) and derived requirements (internal).
6.2 PLANNING A documented decision process shall be used to determine which activities require formal plans, and shall include quality plans applied to projects, functions, products, or organizational entities. Plans shall be kept current and shall include requirements, milestones, responsibilities for performing the work, identification of risks together with the means for addressing them, and controls to be applied.	Project management is responsible for planning, as described in Section 3.4.
6.3 METRICS Metrics to assess conformance to customer requirements shall be developed and used to assure needed corrective actions and improvement measures are taken at the proper time.	Software metrics described in Section 3.1.4: "It is strongly recommended that those who are subject matter experts in the final product be involved in specifying metrics designed to increase product quality and process productivity."
III. PRODUCT QUALITY REQUIREMENTS **1.0 DESIGN DEFINITION** The design agency shall be responsible for design definition of items under its responsibility. Design documents shall incorporate performance requirements and critical characteristics required for the function, reliability, interchangeability, life, and safety of the item. The design and production agencies shall jointly assure that design definition provides all necessary information that requirements are clear, unambiguous, and conform to standard engineering practices. A system for qualifying, approving, and issuing design documents, including changes, shall be established and followed.	Software design requirements are described in Section 3.3.2.1. The description of other program design elements are outside the scope of this deployment document. Traceability from design back to requirements is required. All requirements must be translated into the design document. All software artifacts, including design, are reviewed for conformance to commitments. All software artifacts, including design documents, are subject to change control.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
Design documents shall be maintained in a manner that assures items are procured, manufactured, inspected, tested, and disassembled to the applicable design agency requirements. Procedures and responsibilities shall be established and maintained to control, verify and provide for change to the design of the product to assure that all requirements are met. Complete, current, and accurate records of product definition shall be maintained.	
2.0 INSTRUCTIONS AND PROCEDURES A system which provides and controls documented work instructions for manufacturing, inspection, production and acceptance testing, maintenance, repair, assembly and disassembly shall be established. These instructions shall be available to and followed by the personnel performing the work. The system shall assure that instructions and procedures are adequate, accurate, current, and consistent with design requirements.	Instructions for use of code may be required as part of release. These instructions are subject to review (technical, quality, management). See Section 3.3.3. All artifacts produced are reviewed for consistency with requirements. Other types of instructions are outside the scope of this deployment document.
3.0 DOCUMENT CONTROL A documented system shall be established and maintained to control all documents and data that relate to the requirements of QC-1. The system shall define responsibility for preparing, reviewing, approving, and issuing documents which are adequate, complete and correct. The system shall assure that the latest applicable design documents and change information are released, implemented in a timely manner and specify effectively. In research and development, instructions and procedures may consist of dated and signed notes in a laboratory manual.	Document control is outside the scope of this deployment document.
4.0 PROCUREMENT **4.1 GENERAL** The procurement system shall ensure that purchased product conforms to all specified requirements and that all necessary documentation to establish conformance is provided. DOE reserves the right to perform quality assurance	Procurement is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
surveys and verification inspections at vendor and supplier locations where production materials or services destined for production application are rendered under a contractor's purchase order (contract).	
4.2 PROCUREMENT PLANNING Procurement activities shall be planned and documented to assure a systematic approach to the procurement process. Procurement methods and organizational responsibilities shall be defined. The procurement system shall, as a minimum, address: a. procurement document preparation, review, and control; b. selection of procurement sources; c. bid evaluation and award; d. assessment activities by purchaser; e. control of nonconformance; f. root cause and corrective action; g. acceptance of items or services; h. supplier's calibration program; i. quality records; j. process for controlling and returning defective or nonconforming material to the supplier; and k. quality system.	The description of the procurement planning process is outside the scope of this deployment document.
4.3 SUPPLIER ASSESSMENT The purchaser shall select suppliers on the basis of assessment of ability to meet requirements, including quality requirements. The selection of suppliers shall be based on technical reviews performed by the procuring agency or upon evaluation of historical evidence. Suppliers shall be monitored and evaluated with regard to the effectiveness of their quality system and the quality of their product. The nature and extent of control exercised by the purchaser over the supplier shall depend upon the type of product and the supplier's demonstrated performance.	The description of how suppliers are assessed (supplier assessment) is outside the scope of this deployment document.
4.4 PROCUREMENT DOCUMENTATION Procurement documents shall require the supplier to have an effective quality program. Procurement documents, at all tiers, shall identify documentation, records to be submitted or maintained, and specific retention times. Procurement documents shall provide for access to the supplier's facility and inspection records by the DOE and/or the procuring agency.	Procurement documentation is outside the scope of this deployment document.
4.5 RAW AND COMMERCIAL MATERIAL Raw and commercial materials to be used in processing or manufacturing of product shall be tested to determine conformance to applicable specifications.	Raw and commercial material is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
Unless otherwise required by the purchase order or the product specification, supplier provided test reports may be accepted in lieu of such tests. When supplier provided test reports are used as a basis of acceptance, the test results shall be compared with specification requirements. Proprietary materials may be accepted on label. The validity of supplier provided test reports or labels shall be periodically verified by at least one of the following methods, as appropriate and within the limits of the specification: a. independent testing to requirements determined by the specification; b. auditing; c. testing to typical properties, if verifiable. d. When conditions such as low volume or proprietary processes limit the effectiveness of auditing, independent testing shall be performed within the limits defined by the specification.	
4.6 CERTIFICATE OF CONFORMANCE A certificate of conformance is required for all weapons and weapon related materials and hardware destined for production activities, with the exception of raw and commercial materials. The certificate of conformance must include the following: a. The certificate shall identify the procurement requirements met by the supplier. b. The certificate shall be signed or otherwise authenticated by a person who c. is responsible for this function and whose function and position are described in the supplier's quality assurance program. The certification system, including the procedures to be followed in filling out a certificate and the administrative procedures for review and approval of the certificates, shall be described in the supplier's quality assurance program. Such certifications shall be periodically and independently verified by at least one of the following methods, as appropriate: a. independent testing; b. auditing; c. testing to typical properties, if verifiable.	Certificate of conformance is outside the scope of this deployment document.
5.0 IDENTIFICATION, CONTROL, AND STATUS OF ITEMS Methods shall be established for controlling the identification and status of product throughout the product life cycle until sanitization occurs.	Identification, control, and status of items is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
Status shall be identified by using markings, authorized stamps, tags, labels, routing cards, physical location or other suitable means. Unique tooling and fixtures shall be identified and controlled. Limited life materials/components shall be identified and controlled to preclude use of expired items and provide for efficient recall, if necessary. Controls shall be established for materials designated for destructive testing or special evaluation to prevent inadvertent use/shipment. Instructions for marking and labeling items shall be established as necessary to adequately identify, maintain, and preserve the items, including indication of the presence of special environments or the need for special controls. Software used to maintain material control during automated production, inspection, or disassembly operations shall demonstrate and assure control of materials and material status.	
6.0 CONTROL OF PROCESSES Processes shall be characterized, documented, and maintained under controlled conditions to minimize product/process variability and to prevent nonconformance. Proposed product and process changes throughout the product life cycle shall be evaluated for their potential impact on quality, producibility and maintainability prior to incorporation. Processes, including inspection, test, and acceptance processes, shall be qualified jointly by design and production agencies prior to their use for production and acceptance unless the design agency exempts this requirement. The requirement for production process qualification and characterization may be exempted if production quantities are such that the process will not be repeated, or if inspection and/or testing, including inspection and tests performed on subsequent assemblies, provide adequate assurance of quality.	Control of processes is outside the scope of this deployment document.
6.1 PROCESS CONTROL When production quantities allow, statistical techniques, such as statistical process control, process capability studies, and other preventative measures, shall be utilized to	Process control is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
assure continuous control over production processes and to identify and continually reduce variability. Criteria for workmanship shall be stipulated, to the extent practical, in written standards or by means of representative standards.	
6.2 SPECIAL PROCESSES Special processes shall be identified, and procedures, processes, and controls implemented to assure a high level of confidence in the control of product variability and to minimize nonconformances. Methods shall be established to assure conformance to requirements through qualification and control of equipment, procedures, and/or personnel training. Evidence of certifications/qualifications of personnel, procedures, and equipment shall be maintained.	Special processes are outside the scope of this deployment document.
7.0 INSPECTION, TEST, AND ACCEPTANCE Physical examination, inspection, measurement, or testing of material shall be accomplished under controlled conditions. Measurement uncertainty of the inspection technique shall be considered in the selection of inspection and test equipment or criteria for acceptance or rejection. When measuring and test equipment is used in making measurements for acceptance of product, its evaluation and approval must be documented. When automated manufacturing systems are used as the method of acceptance, they shall be designed, validated, qualified, controlled, and monitored sufficiently to protect product quality such that the completion of the automated operation may be accepted as objective evidence of conformance to requirements. When fixtures, molds, and other such tooling are used as the method of acceptance, they shall be certified prior to release for use. These devices shall be controlled and recertified at established intervals. Product acceptance activities shall be performed to assure compliance to applicable drawings and specifications. When material requires modification, repair, or replacement after product acceptance, there shall be witnessing or verification of the modification, repair, or replacement and reverification of any affected characteristics prior to reacceptance. Sampling plans shall prescribe random sampling unless	Inspection, test, and acceptance as it applies to non-software program elements, is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
otherwise approved by the design agency and shall afford a sound statistical basis to ensure product quality. Test plans for research and development testing programs shall be developed and documented for major activities. The methodologies used to establish test plans shall be adequate to provide confidence in the results.	
8.0 CONTROL OF MEASURING AND TEST EQUIPMENT A standards and calibration program shall be maintained for the purpose of comparing measuring and test equipment with calibration standards of suitable range and accuracy. Standards and measurement devices shall be certified for use in compliance with the requirements of the AL Appendix 56XB, Development and Production Manual, Chapter 8.4.	Control of measuring and test equipment is outside the scope of this deployment document.
9.0 HANDLING, STORAGE, PACKAGING AND DELIVERY Procedures, controls, and facilities shall be maintained to assure that handling, storage, packaging, and shipping operations comply with requirements and prevent damage, deterioration, loss, or substitution.	Handling, storage, packaging and delivery are outside the scope of this deployment document.
9.1 GOVERNMENT FURNISHED MATERIAL Material shipped interproject from one contractor's responsibility to another will be provided as Government Furnished Material. Such DOE accepted material is inspected only for shipping and handling damage by the receiving contractor unless there are valid reasons for requiring additional tests or inspections. Discrepancies noted during assembly or normal handling will be given proper evaluation and disposition. Discrepancies will be reported to the responsible contractor through the DOE.	Government furnished material is outside the scope of this deployment document.
9.2 DOE ACCEPTED MATERIAL Once material is accepted by the DOE, it is considered to be property of the DOE and under its management control. DOE shall be notified when accepted material is issued from stores for purposes different from the original intent. DOE shall also be notified when accepted material is issued to perform additional evaluation, inspection, or rework. The agency shall describe the need for the material and the methods that will be used for processing. Any special handling, storage, processing or evaluation of DOE accepted material must be approved by DOE prior to performance.	DOE accepted material is outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
10.0 CONTROL OF NONCONFORMING ITEMS Procedures shall be established and maintained to ensure that material, which does not conform to requirements, is prevented from inadvertent use, shipment or installation. Control of nonconforming items shall provide for identification, documentation, evaluation, preservation, segregation, and disposition, as well as notification to the organization concerned. There shall be timely disposition of nonconforming material with corrective action and root cause reporting to evaluate possible product or process improvement and any impact on previously produced product and to minimize the probability of recurrence. This activity shall be commensurate with the complexity and the risk associated with failure of the product to meet established requirements. The responsibility for review and the authority for disposition of nonconforming material shall be defined and documented. Nonconforming material may be authorized for "use as is" by the responsible design agency. Repair, rework, or evaluation of nonconforming items shall be performed in accordance with documented procedures approved by the design agency.	Control of nonconforming items is outside the scope of this deployment document.
11.0 CORRECTIVE ACTION Procedures for production related activities shall be established, documented and maintained to: a. determine the root cause of nonconforming product and the corrective action needed to prevent recurrence; b. analyze all processes, work operations, quality records, and reports to detect and eliminate potential causes of nonconformance; c. initiate preventative actions to deal with problems at a level corresponding to the risk encountered; d. apply controls to ensure corrective actions are taken and that they are effective; e. implement and record changes to procedures resulting from corrective action. Any previously produced product with the same conditions shall be identified and disposition shall be provided. Corrective action for research and development operations may be included as noted changes to experiments documented in a laboratory manual consistent with requirements in paragraph 3.0, Section III.	Corrective action is outside the scope of this deployment document.
12.0 RECORDS Documented procedures shall be established and maintained for identification, collection, organization, filing, storage, maintenance, retrieval, distribution, retention	The description of the record system is outside the scope of this deployment document. Software records, as appropriate, will be subject to record system

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
and retirement of records that furnish objective evidence of quality. Records shall be complete, identifiable, and shall be appropriately stamped, initialed, signed and dated by authorized personnel, or otherwise authenticated in order to be considered valid. Authentication may include a statement which clearly identifies the responsible person or organization. Records may be original, copies or electronic. Quality records shall be maintained to demonstrate achievement of the quality requirements and effective operation of the quality system. Pertinent supplier quality records shall be an element of these quality records. All quality records shall be legible and stored such that they are readily retrievable in facilities that provide a suitable environment to minimize deterioration or damage and to prevent loss. Retention shall comply with DOE Order 1324.5B, Records Management Program. Procurement, production, inspection, acceptance testing, repair and disassembly documentation that provides traceability to identify product and its origin shall be maintained. Such records are required to: a. certify material quality and provide substantiating evidence; b. identify materials and components contained in the final product; c. provide identification of production and inspection operations performed on product to help preclude improper processing or use; d. provide for timely recall of suspect product; e. provide data with which to analyze performance problems and take timely corrective action; f. provide identification of disassembly performed to help preclude improper processing or disposition.	requirements as described in Section 3.5.2, Configuration Management.
13.0 AUDITS An assessment program shall be established and documented to independently determine compliance with requirements and verify the effectiveness of the quality system. Assessments shall be performed in accordance with written procedures or checklists. Assessments shall be scheduled on the basis of the status, quality history and importance of the activity and shall be planned to provide coverage and coordination with ongoing quality program activities. Assessment results shall be documented and brought to the attention of personnel having responsibility for the	Software assessments are described in Section 4. The description of other types of program assessments are outside the scope of this deployment document.

QC-1	Sandia National Laboratories ASCI Software Quality Program
QC-1, Revision 9	**Mapping / Tailoring Comments**
area/process assessed. Deficiencies and noncompliance' s identified shall have root cause determination and correction. This activity shall be commensurate with the complexity and the risk associated with failure of the product to meet established requirements. Technical reviews for research and development shall employ design reviews, peer reviews, objective "second looks", or other equivalent methods. These reviews shall be formal, periodic, and utilized as independent assessments. These processes shall be documented.	
14.0 SOFTWARE QUALITY ASSURANCE A software quality assurance program shall be established that provides assurance that software is consistent with applicable specifications.	A definition of Software Verification is in this deployment document.
Error prevention and software engineering principles shall be applied to software acquisition, development, use, and maintenance.	The lifecycle approach applies to all software falling within the scope of this deployment document.
Software quality assurance activities shall be commensurate with the complexity and the risk associated with failure of the software to meet established requirements.	The scope of the GP&G and this deployment document. Graded approach. Software verification testing demonstrates compliance with established requirements.
The program shall include weapon or weapon-related software that: ▪ controls the function of weapon and weapon-related components; ▪ controls design or design verification; ▪ controls production processes or equipment; ▪ controls testing or inspection processes or equipment; ▪ controls calibration of standards and measurement devices; or ▪ provides analysis capability to determine product acceptability. ▪	The scope of the GP&G and this deployment document.
The program shall address all elements of QC-1 as they apply to the software component	The ASCI software V&V program consists of a number of interrelated documents that implement the elements of QC-1 as applied to software. This deployment document implements a subsection of those elements of QC-1 that apply to software per the scope section and tailoring described in the GP&G.

Appendix C: Assessment Checklist

A blank checklist begins on the next page.

Assessment Checklist for ASCI Apps Software Development Areas

(1) Application Name: Application Class: Assessment Date:	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
Practice	3=Fully 2=Partially 1=Plan to	3=Fully 2=Partially 1=Plan to 0=Not addressed NA - not applicable	3=Fully 2=Partially 1=Plan to 0=Not addressed NA - not applicable	Use this area to explain why NA is selected as a response to columns (3) or (4) and to demonstrate evidence for other responses as needed.
Software Engineering				
1. Requirements Phase				
1a. Derive software requirements. *Section 3.3.1*				
1b. Document software requirements. *Section 3.3.1*				
1c. Assess feasibility, if applicable, and generate estimates for budget, resources, etc. *Section 3.3.1*				
1d. Establish acceptance criteria based on requirements. *Section 3.3.1*				
1e. Determine necessary links to other layers of requirements, code, and tests. *Section 3.3.1*				
1f. Ensure requirements traceability to other product artifacts throughout subsequent software phases. *Section 3.3.1*				
1g. Review and approve requirements artifacts. *Section 3.3.1*				
2. Development: Design Subphase				
2a. Derive the design. *Section 3.3.2.1*				
2b. Communicate the design to the team. *Section 3.3.2.1*				

(1) Application Name: Application Class: Assessment Date:	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
2c. Document the design. *Section 3.3.2.1*				
2d. Evaluate impact to requirements. *Section 3.3.2.1*				
2e. Plan for testing: initiate development of test plan. *Section 3.3.2.1*				
2f. Review and approve design artifacts. *Section 3.3.2.1*				
3. Development: Implementation Subphase				
3a. Evaluate impact of implementation to design and requirements. *Section 3.3.2.2*				
3b. Translate design into code and other software product artifacts. *Section 3.3.2.2*				
3c. Communicate issues with requirements/design team and developers. *Section 3.3.2.2*				
3d. Review and approve implementation artifacts. *Section 3.3.2.2*				
4. Development: Test Subphase				
4a. Finalize test plan. *Section 3.3.2.3*				
4b. Execute test cases found in test plan. *Section 3.3.2.3*				
4c. Review test case output using acceptance criteria defined in test plan. *Section 3.3.2.3*				
4d. Document test case results. *Section 3.3.2.3*				
4e. Retest updated software if acceptance criteria is not satisfied. *Section 3.3.2.3*				

(1) Application Name: Application Class: Assessment Date:	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
4f. Review and approve Test Subphase outputs. *Section 3.3.2.3*				
5. Release Phase				
5a. Receive and evaluate release request. *Section 3.3.3*				
5b. Plan and develop release. *Section 3.3.3*				
5c. Review and approve release. *Section 3.3.3*				
5d. Create and distribute release. *Section 3.3.3*				
5e. Support release, as agreed with customer. *Section 3.3.3*				
Project Management				
6. Project Planning				
6a. Submit IP addressing project tasks annually. *Section 3.4.1*				
7. Tracking and Oversight				
7a. Review milestone status quarterly. *Section 3.4.2*				
7b. Issue Baseline Change Proposals (BCPs), if needed. *Section 3.4.2*				
7c. Prepare performance reports on a quarterly basis. *Section 3.4.2*				
8. Risk Management				
8a. Incorporate risk identification and risk mitigation into project execution using the BCP. *Section 3.4.3*				
Support Elements				
9. Requirements Management				
9a. Conduct requirements tracing. *Section 3.5.1*				

(1) Application Name: Application Class: Assessment Date:	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
9b. Determine requirements ownership and status tracking. *Section 3.5.1*				
10. Configuration Management				
10a. Conduct issue tracking of software product artifacts, including requirements. *Section 3.5.2*				
10b. Perform version control of software product artifacts, including requirements. *Section 3.5.2*				
10c. Perform release and distribution management. *Section 3.5.2*				
10d. Engage in ASCI records management. *Section 3.5.2*				
11. Third Party Software				
11a. Accept third party software and libraries into the application code domain. *Section 3.5.3*				
11b. Install, integrate, & control the accepted third party software. *Section 3.5.3*				
12. Training				
12a. Train appropriate project members in use of project management and project tracking and oversight processes. *Section 3.5.4*				
12b. Train staff on activities necessary for producing software artifacts. *Section 3.5.4*				
12c. Train staff on use of software tools. *Section 3.5.4*				
12d. Train staff on software processes and their implementation. *Section 3.5.4*				

(1) Application Name: Application Class: Assessment Date:	(2) AQMC Requires:	(3) Code Team Evaluation:	(4) Assessment Team Evaluation:	(5) Comments/Evidence for Code Team or Assessment Team
12e. Train staff on software verification process and techniques. *Section 3.5.4*				
Total Number of Areas	**12**			
Total Number of Practices	**46**			
(6) Completed By: (print name and date) (signature)	---------- ----------			