SAND2003-1899
Unlimited Release
Printed June 2003 (V 1.0)

Trilinos Developers Guide
OBSOLETE
See TrilinosSQE/Plans/DevGuide2.doc
Part II: ASCI Software Quality Engineering Practices
Version 2.0

Michael A. Heroux and James M. Willenbring
Computational Mathematics & Algorithms Department

Robert Heaphy
Discrete Algorithms and Math Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185

Abstract

The Trilinos Project is an effort to facilitate the design, development, integration and ongoing support of numerical software libraries, primarily focused on solvers. A new software capability is introduced into Trilinos as a *package*. A Trilinos package is an integral unit and, although there are exceptions such as utility packages, each package is typically developed by a small team of experts in a particular algorithms area such as algebraic preconditioners, nonlinear solvers, etc.

This second part of the Trilinos Developers Guide is a resource for Trilinos package developers who are working under the Advanced Scientific Computing Initiative (ASCI) and are therefore subject to the ASCI Software Quality Engineering Practices as described in ASCI Applications Software Quality Engineering Practices document [1]. The Trilinos Developers Guide [2] is a companion document to this second part and contains much of the detailed information that is essential for all Trilinos developers.

Intentionally Left Blank

1. Introduction

One objective of the Advanced Scientific Computing Initiative (ASCI) is to develop professional software quality engineering (SQE) practices that will ensure the quality of application software developed with ASCI funding. To this end, the ASCI Applications Software Quality Engineering Practices document [1], lists 47 practices that should be address by application and library software developers. Part II of the Trilinos Developers Guide addresses each of the 47 practices, often referring to the main Trilinos Developers Guide [2] except for issues that are unique to ASCI SQE practices. Each of the 47 practices is discussed in terms of the responsibilities of the Trilinos Framework and each individual Trilinos Package. As can be seen from the table in Section 4, the Trilinos Framework provides a valuable service to the Trilinos Packages, providing package developers with ready-made support for many of the 47 practices, leaving to package developers only those practices that should be under the direction of each package team. In fact, 32 of the practice are the sole responsibility of the framework and the framework provides significant support for the remaining 15 practices.

2. Roles, Documents, Tools and Events

The primary content of this document is a large table in Section 4, listing and discussing each of the 47 ASCI SQE practices. Throughout the discussion, a number of roles (people), documents, tools and events are cited frequently. We list each of them here and assign acronyms where appropriate:

Roles:

- **ASCI Program Management:** The ASCI program (which includes ASCI Algorithms) has an evolving set of process for SQE. Many of the SQE processes in the Requirements, Project Planning, Tracking and Oversight and Risk Management phases are driven by decisions made by ASCI Program Management.
- **ASCI Algorithms Principal Investigator:** The Trilinos Project receives a significant portion of its funding from the Algorithms portion of the ASCI Program. As a result, the ASCI Algorithms Principal Investigator plays an important role in the Requirements, Project Planning, Tracking and Oversight and Risk Management phases for Trilinos, following the guidelines and requirements established by ASCI Program Management.
- **Trilinos Project Leader:** A fundamental management principal of the Trilinos Project is that packages should be as autonomous as possible, recognizing that local control with careful attention to interfaces is often the most effective approach to producing high quality software. At the same time there is a need for a single focal point in some situations. The Trilinos Project leader is the primary focal point for major project decisions. The duties of this person include:

1. Arbitrating in cases where a consensus decision cannot be reached as part of inter-package decisions.
2. Organizing Trilinos project events (see below).
3. Managing and tracking progress on the Trilinos Framework Responsibilities as described in Section 4.

- **Trilinos Release Manager:** During the release process, the Trilinos Release Manager is responsible for coordinating the package versions that will be used for the Trilinos release, and for tagging and branching the repository as part of the release.
- **Trilinos Capability Leaders:** Because of the broad scope of Trilinos, we have leaders assigned to the following capability areas:
  1. Framework, Tools & Interfaces.
  2. Discretizations.
  3. Geometry, Meshing & Load Balancing.
  4. Scalable Linear Algebra.
  5. Linear & Eigen Solvers.
  6. Nonlinear, Transient & Optimization Solvers.

  These leaders have responsibilities across packages and are responsible for strategic planning for Trilinos in their areas.
- **Package Leader:** Each Trilinos package has one or more leaders. These leaders are responsible for managing and tracking package responsibilities as described in Section 4. These leaders are also responsible for attending the Monthly Trilinos Leaders Meetings, representing the package development team and disseminating information to the team.
- **Package Developer:** Each package has an identifiable group of developers. Any given individual may be a member of multiple package development teams.

Documents:

- **Trilinos Developer Guide (TDG):** Primary development guide for Trilinos developers. Discusses software requirements that apply to all Trilinos developers and presents the suggested practices for each requirement. This guide also describes the services available to packages that are part of Trilinos [2].

- **ASCI Algorithms Annual Plan (AAAP):** Each fiscal year the ASCI Algorithms team gathers user and software requirements from ASCI application teams and its own members. The exact form and number of documents generated has changed from year to year, as the ASCI program itself defines its processes for software quality. In past years we have provided all analysis and documentation request by ASCI program managers, and have developed additional documents that provide greater detail than what ASCI requires. As ASCI program practices evolve, we will continue to adapt our documents and processes to match, especially

Key point: Although the type and number of documents required by the ASCI program has and will continue to change, for simplicity we refer to entire collection of these documents as the ASCI Algorithms Annual Plan (AAAP).

in the Requirements, Project Planning, Tracking and Oversight and Risk Management phases. Although the type and number of documents required by the ASCI program has and will continue to change, for simplicity we refer to entire collection of these documents as the ASCI Algorithms Annual Plan. In the recent past, the primary documents have been the ASCI Algorithms Implementation Plan and the ASCI Algorithms Objectives and Resource Plan.

- **ASCI Algorithms Quarterly Report (AAQR):** At the end of each quarter, when requested by ASCI program management, the ASCI Algorithms team will generate a progress report and list adjusted milestones as needed.

- **Package Developer Guide (PDG):** In the Package Responsibilities column we discuss in detail how a package should satisfy a particular practice. However, it is always the case that a package may satisfy any given practice via its own process as long as the alternate process is documented in the appropriate Package Developer Guide. For example, if Package X derives its design by an alternative process than what we describe below for Practice 2a, then there should be a Package X Developer Guide and it should document the practice used to satisfy Practice 2a.

Key point: It is always the case that a package may satisfy any given practice via its own process as long as the alternate process is documented in the appropriate Package Developer Guide.

Tools:

- **Concurrent Versions Systems (CVS):** All Trilinos source code and documents are maintained using CVS [3]. The primary CVS repository resides on the Trilinos Development platform "software.sandia.gov" on the Sandia Open Network (SON). Sensitive documents are retained under a separate repository on the Sandia Restricted Network (SRN). Use of CVS is documented in the TDG.
- **Bugzilla:** All major features and software faults are reported using Bugzilla [4], a web-based issue-tracking package. Each Trilinos package, the Trilinos framework and Bugzilla itself are set up as Bugzilla products. Bugzilla is available at http://software.sandia.gov/bugzilla. Use of Bugzilla is documented in the TDG.
- **Mailman:** Each Trilinos package has a set of Mailman [5] mail lists to support communication and archiving of important information and artifacts. List descriptions are documented in the TDG.
- **Doxygen:** Many Trilinos packages use the documentation-generating package called Doxygen [6]. Doxygen processes source code comments producing detailed online and printed documentation of the processed source. Although it can be used with many language and is highly configurable, the most common use of it within Trilinos is to process header files containing description of C++

classes and detailed documentation of the major user-callable methods in each class. Doxygen also extracts information about class interactions and dependencies. Most Trilinos packages presently use Doxygen to provide user reference documentation.

- **Microsoft Project:** The Trilinos project manager uses Microsoft Project [7], a project management and tracking tool, to track and communicate major Trilinos deliverables for Class A Trilinos packages. See Section 3 for a description of Trilinos package classification.
- **Autoconf, Automake, and Libtool:** Trilinos configuration and building is facilitated by using Autoconf [8], Automake [9] and Libtool [10], collectively referred to as Autotools [11]. These tools facilitate dynamic configuration and building of Trilinos across a broad set of computer platforms. Via runtime compilation and linking tests, queries to the operating system and user-specified parameters, Trilinos can be configured and built on almost any platform with minimal user knowledge of details such as location of system libraries and compilers. The autotools build system also supports installation of Trilinos for multiple users and automatic creation of distribution tar files.
- **software.sandia.gov**: The primary Trilinos development platform is a Linux server called software.sandia.gov. This platform supports all of the tools listed above, contains the Trilinos CVS repository and all of the Mailman and Bugzilla archives. This platform is on the Sandia Open Network (SON), so it is accessible from any internet-connect machine. The file systems on this platform are backed up daily. Backup tapes are shipped offsite monthly.

Events:

- **Monthly Trilinos Leaders Meeting:** Package leaders for Trilinos packages, Trilinos management and other stakeholders participate in a monthly leadership meeting. Meeting minutes are sent to the Trilinos-Leaders@software.sandia.gov mail list for communication and archiving. Meeting topics include discussion of requirements, design, implementation, testing and documentation. We also conduct developer training as needed during these meetings. A meeting agenda is sent to the Trilinos-Leaders mail list prior to each meeting.
- **Annual Trilinos User Group Meeting:** Approximately once a year we hold a meeting for Trilinos users. During this meeting we present an overview of Trilinos, detailed presentations of Trilinos packages. The tentative date for the first of these meetings is October 20, 2003.

3. Trilinos Framework and Packages Class Categories.

The ASCI Software Quality Engineering Practices identifies three classes of software products: A, B and C. At this time, the Trilinos Framework itself is considered Class A, as are the following packages:

- Epetra.

- AztecOO.
- Ifpack.

Numerous other Trilinos packages are under development at this time, including NOX, Anasazi, ML, etc. However, these packages are presently considered Class B products, since the primary focus of the work in these packages is still research and development. At a time when a package becomes part of a production application, it will be reclassified as Class A.

4. Trilinos Practices Table

The remainder of this document is a table that follows, item by item, the 47 practices listed in the ASCI Software Quality Engineering Practices document. For each practice, the responsibilities of the Trilinos Framework and each Trilinos Package are described using present-tense phrasing. Please note that some of these responsibilities are not fully addressed at this time, in which case this document serves as a plan rather than a statement of practice.

Key point: … some of these responsibilities are not fully addressed at this time, in which case this document serves as a plan rather than a statement of practice.

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
Software Engineering		
1. Requirements Phase		
1a. Gather user requirements	Primary user requirements are gathered and documented in the AAAP. In addition, topics discussed during the monthly Trilinos Leaders meeting include Trilinos user requirements. Meeting minutes are archived on the trilinos-leaders@software.sandia.gov mail list.	None.
1b. Derive software requirements.	Primary software requirements are gathered and documented in the AAAP.	None.
1c. Document software requirements.	Software requirements are documented in the AAAP.	None.
1d. Assess feasibility, if applicable, and generate estimates for budget, resources, etc.	Feasibility, budgets and resources are determined during development of the AAAP.	None.
1e. Establish acceptance criteria based on requirements.	Acceptance criteria are stated in the AAAP.	None.

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
1f. Determine necessary links to other layers of requirements, code, and tests.	Any links are stated in the AAAP.	None.
1g. Ensure requirements traceability to other product artifacts throughout subsequent software phases.	Requirements traceability is accomplished via the AAAP and the AAQR.	None.
1h. Review and approve requirements artifacts.	All principal investigators review and approve the AAAP and AAQR. Approval is indicated by email sent to the Trilinos-developers mail list.	None.
2. Development: Design Subphase		
2a. Derive the design.	Doxygen [6] is provided on the Trilinos Development platform software.sandia.gov.	Package Developer derives the design via a Doxygen-commented header file containing methods with little or no executable code.
2b. Communicate the design to the team.	A Mailman mail list called package-developers@software.sandia.gov is provided for each Trilinos package.	Package Developer generates Doxygen HTML documentation and sends the documentation to the package-developers@software.sanda.gov mail list, where other package developers see and comment on the design, and where the documentation is retained in a Mailman archive.
2c. Document the design.	A link from the main Trilinos web page http://software.sandia.gov/trilinos is provided for each package.	The design is automatically documented via the Doxygen-commented header file. The design is available via the Mailman list archives at http://software.sandia.gov/mailman/listinfo. When the package is available for download, the Doxygen-generated documentation will also be available from the Trilinos website http://software.sandia.gov/trilinos.
2d. Evaluate impact to requirements.	The AAAP and AAQR will be updated as necessary if any requirements need to be modified as a result of design issues.	The Package Developer will notify the ASCI Algorithms Principal Investigator of any design issues that may impact requirements.
2e. Plan for testing: initiate development of test plan.	An automatic testing harness is provided so that any test scripts checked in to specific directories in the Trilinos repository are automatically dispersed to a collection of test platforms and run on a nightly basis, with test output being sent to a regression mail list for archiving, and also to the test script owner.	Package Developer saves unit testing and any example drivers in the Trilinos CVS repository and places test scripts into the appropriate predefined directories for inclusion in the Trilinos automated regression tests. This process is documented in the TDG.
2f. Review and	A Mailman mail list called package-	The development team, via the package

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
approve design artifacts.	developers@software.sandia.gov is provided for each Trilinos package.	developers mail list, reviews a design. The design is approved when consensus has been reached by discussion on the package developers list and the primary feature developer has sent a follow-up note indicating this consensus. In instances where consensus cannot be reached, the package leader will arbitrate.
3. Development: Implementation Subphase		
3a. Evaluate impact of implementation to design and requirements.	The monthly Trilinos Leaders Meeting includes discussion of requirements, design and implementation. Any changes to design or requirements are reflected in the AAAP and AAQR.	None.
3b. Translate design into code and other software product artifacts.	A CVS repository is provided for all code and artifacts.	All source files related to a given package are maintained in the Trilinos CVS repository under the directory Trilinos/packages/package_name
3c. Communicate issues with requirements/design team and developers.	A Mailman mail list called Trilinos-developers@software.sandia.gov is used for interpackage development discussions. In addition, a list called package-developers@software.sandia.gov is provided for each Trilinos package. All email discussions are archived. Email archives are browsable and searchable using a web browser.	All major design and requirements discussions are summarized and sent to the trilinos-developers list or to the package-developers list, or are conducted via email discussion using this list.
3d. Review and approve implementation artifacts.	A Mailman mail list called package-developers@software.sandia.gov is provided for each Trilinos package. A bugzilla product is provided for each package.	As a feature nears completion, it must pass all Trilinos regression tests. Also, Doxygen documentation for each user-level class must be completed and the feature must be portable to all supported platforms. Review of a major feature involves the package developers, the primary customer asking for the feature and the Trilinos build support representative. The implementation is approved when all of these parties agree it is complete. The Bugzilla issue report for each package feature is closed upon approval.
4. Development: Test Subphase		
4a. Finalize test plan.	The Trilinos framework test plan is as follows: Prior to the release of a version of Trilinos, a certification run of the full Sierra test suite will be executed with the release candidate. Success or failure of the test suite is recorded on the Trilinos-regression mail list. In addition, the TDG discusses	Package developers provide full unit and integration tests, checking driver scripts into the appropriate queue directories in the Trilinos CVS repository. Package developers run these tests prior to checkin of code to CVS. These tests also serve as a regression test suite via the Trilinos

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
	the automatic regression testing services and the plan for executing tests. Depending on the directory to which a test scripts belongs, a test will run daily or weekly, and in serial or MPI mode.	automated test harness.
4b. Execute test cases found in test plan.	The Trilinos automatic regression test harness runs nightly across a broad set of supported platforms.	None.
4c. Review test case output using acceptance criteria defined in test plan.	Test results from automated test harness are sent to the package-regression mail list and also to the email address of the script owner. The subject line of the email indicates if tests passed or failed. If one or more tests failed, the verbose output from the test is attached the email message.	Package developers provide two modes of execution for each test script. The first mode is a silent return with an exit status of pass or fail (See the TDG for details). Passing a "-v" argument to the test script, in which case the test will run in verbose mode, providing detailed information on the test results, activates the second mode.
4d. Document test case results.	Test case results are automatically mailed to the appropriate list package-regression@software.sandia.gov. They are archived on this list.	None.
4e. Retest updated software if acceptance criteria are not satisfied.	Test cases are run nightly. Test cases are also run manually on the release candidate prior to release.	None.
4f. Review and approve Test Subphase outputs.	Test output is reviewed by the Trilinos Project leaders, who subscribes to the Trilinos-regression mail list. Test output is approved when all regression tests pass. Prior to release, Sierra tests are executed and email approval is sent by the Sierra code team representative to the Trilinos project leader, and is copied to the Trilinos-developers mail list.	None.
5. Release Phase		
5a. Receive and evaluate release request.	Release requests are negotiated between Trilinos project leader and customers. A release cycle commences with the announcement of a release target date to the Trilinos-developers mail list.	None.
5b. Plan and develop release.	The above-mentioned email message also describes the release plans, or the plan is discussed as part of the monthly Trilinos Leaders meeting.	Package developers notify the Trilinos release manager about which version of each package should be included in the Trilinos release. Often this is the main development branch, but it can also be a special tagged version of the package source.
5c. Review and approve release.	Email is sent by the Sierra code team representative to the Trilinos project leader, and is copied to the Trilinos-developers mail list. Upon successful testing and review of release candidate, the Trilinos	None.

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
	project leader approves the release.	
5d. Create and distribute release.	A compressed tar file containing the Trilinos source code is generated via the Trilinos autotools process and sent to the Trilinos web server. Access is provided at the Trilinos download page. An announcement of the release is sent to the Trilinos-Announce mail list.	None.
5e. Support release, as agreed with customer.	All software issues are tracked via the Trilinos bugzilla site. In addition, a list of Frequently Asked Questions (FAQs) is maintained on the Trilinos web site.	Package developers must respond to Bugzilla bug reports in a timely manner.
Project Management		
6. Project Planning		
6a. Submit IP addressing project tasks annually.	The AAAP is written and submitted annually.	None.
7. Tracking and Oversight		
7a. Review milestone status quarterly.	The AAQR is written and submitted quarterly, when requested by ASCI program management.	None.
7b. Issue Baseline Change Proposals (BCPs), if needed.	Any BCPs are included in the AAQR, when and as requested by ASCI program management.	None.
7c. Prepare performance reports on a quarterly basis.	Performance reports are included in the AAQR, when and as requested by ASCI program management.	None.
8. Risk Management		
8a. Incorporate risk identification and risk mitigation into project execution using the BCP.	Risk identification and mitigation is included in the AAAP and AAQR, when and as requested by ASCI program management.	None.
Support Elements		
9. Requirements Management		
9a. Conduct requirements tracing.	The Trilinos Project Leader traces requirements via Microsoft Project.	None.
9b. Determine requirements	The Trilinos Project Leader determines requirements ownership and tracks their	None.

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
ownership and status tracking.	status using Microsoft Project.	
10. Configuration Management		
10a. Conduct issue tracking of software product artifacts, including requirements.	A Bugzilla product is provided for each Trilinos package. All issues are tracked via Bugzilla and the underlying MySQL[12] database. Bugzilla provides a full-featured search capability.	Package developers, or their customers, file issue reports using the Trilinos Bugzilla site. This includes major features requests, and software problems. Issue reports are kept up-to-date.
10b. Perform version control of software product artifacts, including requirements.	A CVS repository is maintained for all Trilinos packages. Package-checkins mail lists archive all product modifications. Bugzilla tracks and archives all requirements.	Package Developers utilize the Trilinos CVS repository and the Bugzilla issue tracking software.
10c. Perform release and distribution management.	All Trilinos releases are tagged and branched in the Trilinos CVS repository. We actively support 1 or more release branches in addition to our primary development branch.	None.
10d. Engage in ASCI records management.	Until an identifiable ASCI records management process is provided, the Trilinos project follows the Sandia records management process. The Trilinos development platform (software.sandia.gov) is backed up regularly and backup tapes are shipped offsite. All CVS, Mailman and Bugzilla data and artifacts are retained indefinitely.	None.
11. Third Party Software		
11a. Accept third party software and libraries into the application code domain.	Supported versions of all third party software are kept in the Trilinos3PL CVS repository. These packages are tested regularly via the Trilinos packages that interface with each third party software package.	None.
11b. Install, integrate, & control the accepted third party software.	In addition to retaining third party software in the Trilinos3PL repository, this software can be integrated using the Trilinos autotools configuration parameters as documented in the TDG. Testing of third party software is included in standard unit, integration and regression tests. The Trilinos3PL archive is tagged and matched with each release of Trilinos. Detailed information about the compilers, libraries and other third party software used to build Trilinos is captured in the config.log file in the Trilinos build directories.	None.

Practice	Trilinos Framework Responsibilities	Trilinos Package Responsibilities
12. Training		
12a. Train appropriate project members in use of project management and project tracking and oversight processes.	Training is performed as needed during the Monthly Trilinos Leaders Meeting. Training events are announced on the meeting agenda that is sent to the Trilinos-Leaders mail list prior to the meeting. All package leaders, or designated representatives, are expected to attend. Documentation of training is recorded in the meeting minutes sent to the Trilinos-leaders mail list.	None.
12b. Train staff on activities necessary for producing software artifacts.	Training is performed as needed during the Monthly Trilinos Leaders Meeting. Training events are announced on the meeting agenda that is sent to the Trilinos-Leaders mail list prior to the meeting. All package leaders, or designated representatives, are expected to attend. Documentation of training is recorded in the meeting minutes sent to the Trilinos-leaders mail list.	None.
12c. Train staff on use of software tools.	Training is performed as needed during the Monthly Trilinos Leaders Meeting. Training events are announced on the meeting agenda that is sent to the Trilinos-Leaders mail list prior to the meeting. All package leaders, or designated representatives, are expected to attend. Documentation of training is recorded in the meeting minutes sent to the Trilinos-leaders mail list.	None.
12d. Train staff on software processes and their implementation.	Training is performed as needed during the Monthly Trilinos Leaders Meeting. Training events are announced on the meeting agenda that is sent to the Trilinos-Leaders mail list prior to the meeting. All package leaders, or designated representatives, are expected to attend. Documentation of training is recorded in the meeting minutes sent to the Trilinos-leaders mail list.	None.
12e. Train staff on software verification process and techniques.	Training is performed as needed during the Monthly Trilinos Leaders Meeting. Training events are announced on the meeting agenda that is sent to the Trilinos-Leaders mail list prior to the meeting. All package leaders, or designated representatives, are expected to attend. Documentation of training is recorded in the meeting minutes sent to the Trilinos-leaders mail list.	None.

References

[1] J. Zepper, K Aragon, M. Ellis, K. Byle and D. Eaton, *Sandia National Laboratories ASCI Applications Software Quality Engineering Practices, Version 2.0*, Sandia National Laboratories, SAND2003-XXXX, January 2003.

[2] M. Heroux, J. Willenbring and R. Heaphy, *The Trilinos Developers Guide, Version 1.0*, Sandia National Laboratories, SAND2003-XXXX, May 2003.

[3] Gnu CVS Home Page: http://www.gnu.org/software/cvs.

[4] Mozilla Bugzilla Home Page: http://www.mozilla.org/projects/bugzilla.

[5] Mailman Home Page: http://www.gnu.org/software/mailman.

[6] Doxygen Home Page: http://www.doxygen.org.

[7] Microsoft Project Home Page: http://www.microsoft.com/office/project.

[8] Autoconf Home Page: http://www.gnu.org/software/autoconf.

[9] Automake Home Page: http://www.gnu.org/software/automake.

[10] Libtool Home Page: http://www.gnu.org/software/libtool.

[11] G. Vaughan, B. Elliston, T. Tromey, and I. Taylor. *GNU Autoconf, Automake and Libtool*.New Riders, 2000.

[12] MySQL Home Page: http://www.mysql.com.