The Trilinos Software Lifecycle Model

James M. Willenbring, Michael A. Heroux, Robert T. Heaphy
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185
{jmwille, maherou, rheaphy}@sandia.gov

Abstract

The Trilinos Project [1, 2] is an effort to facilitate the design, development, integration and on-going support of mathematical solver libraries. Efforts range from research and development of new algorithms to proof-of-concept of new and existing algorithms to eventual production use of solver libraries on a variety of computer systems across a broad set of applications. Software quality assurance and engineering (SQA/SQE) play an integral role in the project. Although many formal software lifecycle models exist, no single model can address all Trilinos developer needs since our requirements for rigor change as a particular Trilinos package matures. In this report we present a three-phase promotional lifecycle model that closely matches the needs and realities of Trilinos development.

1. Introduction

A major component of any software project is a software lifecycle. Whether a model is formally defined, or the lifecycle occurs in an *ad hoc* fashion, it does exist. In this document we define a lifecycle model, really a meta-model, that accurately captures the reality of our software engineering environment.

Although many formal software lifecycle models exist, the environment in which Trilinos software is developed is somewhat unique and challenging when compared to the large body of commercial software environments. On the one hand, we are tasked to develop algorithms and software that are leading-edge, with the goal of solving problems that were previously intractable. On the other, we are required to deliver software that can eventually be used to certify critically important engineering systems.

An added dimension is that Trilinos is composed of *packages*: self-contained pieces of software that are developed by semi-independent small teams. Each package matures at its own pace, typically evolving from a small algorithms study project to becoming a widely-used piece of software, embedded in multiple applications. Furthermore, the requirements for rigor change as a particular Trilinos package matures.

In this paper we present a three-phase promotional lifecycle model that we believe closely matches the needs and realities of Trilinos developers. It allows small algorithms-focused efforts to develop in a dynamic, customer-interactive environment while encouraging more mature packages to drive toward a fully-certified software environment that can withstand the rigorous requirements necessary for production computing.

As is true with many SQA/SQE issues, Trilinos packages are allowed to define individualized lifecycles. However, most packages choose not to. By default, packages adopt the three-phase promotional lifecycle model described below. The Trilinos Software Lifecycle Model recognizes the natural transition that most Trilinos packages follow from being a research project to a production quality code. Different packages are expected to be at different points along this lifecycle model. The three phases are:

1) Research
2) Production Growth
3) Production Maintenance

Each phase contains a different lifecycle model and different required processes. Moving from one phase to another is facilitated by a *promotional event*. This model is independently applied to different major versions of Trilinos packages. Below is a description of what is required in each phase and what constitutes a promotional event.

29th International Conference on Software Engineering Workshops(ICSEW'07)
0-7695-2830-9/07 $20.00 © 2007 IEEE

2. Lifecycle Phase 1: Research

In this first phase of development, conducting research is the primary goal; producing software is potentially incidental to research. Any software that is produced is typically a "proof of concept" or prototype. Software that is in this phase may only be released to selected internal customers to support their research or development and should not be treated as production quality code.

2.1. Phase 1 Required Practices

The required practices and processes for this phase are appropriate for efficient research. A list of these practices and processes can be found in the Trilinos Developer Guide Part II [3]. Here are a few important points:

1) The research proposal is the project plan.
2) Software is placed under configuration control as needed to prevent loss due to disaster.
3) Peer reviewed published papers are primary verification and validation.
4) The focus of testing is a proof of correctness, not software.
5) Periodic status reports should be produced.
6) A lab notebook, project notebook, or equivalent is the primary artifact.

Note that software in the research phase need not be written in the "target" language, or support all target machines, and usually has limited error checking and recovery.

In the research phase, the risk is low (risks are primarily technical and not mitigated by formality of processes). The level of formality is low.

2.2. Phase 1 to Phase 2 Promotional Event

When a package reaches a point at which it is ready to move from a research to production phase, which often takes place when package developers are ready to begin ramping up in preparation for a release, a (possibly virtual) meeting takes place involving package developers, and possibly other stakeholders. At this meeting a number of issues are covered including:

Risk Assessment:

1) What are the package's primary technical and project management risks?
2) How can these risks be mitigated?

Gap Analysis:

1) Which practices and processes must be added or improved to get the package into a releasable state?
2) What special actions or training will be required?
3) What is the target date for complying with the level of practices and processes required for release?

Promotion Decision:

1) Considering the results of the risk assessment, gap analysis, and other data, will the package be promoted to Phase 2?
2) What is the target date for releasing the package?

After the meeting, minutes should be sent to the package developer mail list and the Trilinos-framework list. Minimally, these minutes should provide answer to the questions listed above and note other important topics that were discussed.

3. Lifecycle Phase 2: Production Growth

The goal of the Production Growth phase is to elevate the package to a releasable product, with the eventual goal of satisfying the Advanced Scientific Computing (ASC) Software Quality Plan [4], at a minimum. More specifically, in this phase the goal is to make the software product is suitable for use by highly skilled users.

3.1. Phase 2 Required Practices

The second phase of development requires both a larger number of practices and processes and an increased level of formality, while maintaining a flexible development environment. The required practices and processes can be found in the Trilinos Developer Guide Part II. Here are a few important points:

1) Agile methods (with associated lifecycles) are encouraged, for example the practices and processes promoted by Extreme Programming [5].
2) All essential ASC SQE practices performed at an appropriate level (predetermined

during promotion event from the research phase).
3) Artifacts should naturally "fall out" from SQE practices and periodic status reviews and management reports.
4) Process improvement and metrics are appropriate.

This phase may be cyclic (spiral, etc.) as new algorithms become incorporated.
The software may not yet support all intended missions or platforms.

The risk level is medium (the technical risks are reduced and the total risk is more project management oriented such as schedule, budget, staffing, etc.). The default level of formality is medium.

3.2. Attaining Releasable Status

Once the required level of practices and processes has been met for Phase 2, or a waiver has been obtained from the Trilinos Project Leader for any items that do not apply (all or in part) to a particular package, an email indicating that the package is ready for release is sent to the trilinos-framework list. The email should include attachments or links with artifacts supporting that each of the required practices and processes have been implemented for the package. The Trilinos Project Leader or a designee will review the artifacts and determine if the package is ready for release as a part of Trilinos. If not, that individual will ask for additional artifacts for any practice or process that does not appear to have sufficient documentation. After the set of artifacts has been approved, the package will be eligible for release as a Trilinos package beginning with the next release.

3.3. Phase 2 to Phase 3 Promotional Event

After a package has been under long-term development, and is reaching a point where its features are mature, it becomes necessary to provide a more complete set of artifacts than are typically produced by Agile methods. In order to facilitate final post-delivery maintenance, especially for the case where the original development team is not longer available to support the package, we provide a third phase.

The promotional event elevating a package to Phase 3 should be triggered by a point in the project when the focus of modifications turns from new development to minor enhancements and bug fixes. At this point, it is appropriate to prepare for the long-term future of the package. This promotional event will be initiated by the development team, a manager, or possibly by the Trilinos Project Leader. The event itself is again a (possibly virtual) meeting involving package developers and optionally other stakeholders including members of management, customers, other Trilinos developers, or potential future package maintainers.

Several issues need to be addressed at the meeting including:

Risk Assessment:

1) What are the package's primary technical and project management risks?
2) How can these risks be mitigated?

Gap Analysis:

1) Which practices and processes must be added or improved to get the package into a maintenance ready state?
2) What is the target date for complying with the required level of practices and processes?

Promotion Decision:

1) What is the medium to long-term funding outlook for the package?
2) Who is going to provide long-term maintenance services for the project? (One or more of the original developers, or a different group?)
3) If funding is not likely to be available for future maintenance, current customers should be notified so they have a chance to offer continued funding if it is in their best interest. A list of these customers should be produced.
4) If the customer base of the package is small or the package has been replaced with another code, the development team may consider retiring the code rather than moving to the third development phase. Any such decision should be approved by customers and management.
5) Considering the answers to the above questions, and other available data, will the package be promoted to Phase 3?
6) What is the target date (if any) for turning the package over to the long-term maintenance team?

Note that packages that have been retired without attaining the level of practices and processes required for a maintenance ready state should not be included in future Trilinos releases.

After the meeting, minutes should be sent to the package developer mail list and the trilinos-framework list. Minimally, these minutes should provide answer to the questions listed above and note other important topics that were discussed.

4. Lifecycle Phase 3: Production Maintenance

The goal of the third and final phase of development is robust software suitable for typical end uses. At this point, the requirements and a prototype software foundation are stable. However, the Agile methods that served well in Phase 2 are not sufficient to support product maintenance during the coming decades of software use where typically only adaptive maintenance, in response to computer system changes, will be performed. A more complete set of artifact is required, and the software itself will require changes to improve maintainability. In the extreme case, it may even make sense to rewrite large portions of the package.

4.1. Phase 3 Required Practices

The full set of required practices and process for Phase 3 can be found in the Trilinos Developer Guide Part II. We include important highlights here.

1) After achieving maintenance ready status, the package may (as determined during the promotion event) be handed over to another party during this phase for continued support and development.
2) If the code is transferred to a different party, the ownership of the design is not necessarily transferred. The design owner must attend meetings concerning requirements changes and potential design changes.
3) A widely-accepted lifecycle model such as the Waterfall or Unified Process [6] methods is used.
4) End of life planning is a key component during this phase. In particular, the software must have good compliance with SQE practices and internal documentation must be formal (UML is suggested).
5) SQE practice compliance and solid documentation will help to ensure a successful transfer of the code, and must be completed whether a transfer is currently planned or not.

The risk level is low (almost totally project management risks, which can be mitigated by appropriate process formality). The default level of formality is high (particularly if the project may be handed off to another party).

4.2. Achieving Maintenance Ready Status

Implementing the required level of practices and processes for Phase 3 means that the package has achieved maintenance-ready status. In particular, the package must support all formal missions and required platforms, be “user-proof”, and user support (training, documentation, “bug reporting” and help desk) must be available.

Once the required level of practices and processes has been met for Phase 3, or a waiver has been obtained from the Trilinos Project Leader for any items that do not apply (all or in part) to a particular package, an email is sent to the trilinos-framework list. The email should include attachments or links with artifacts supporting that each of the required practices and processes have been implemented for the package. The Trilinos Project Leader or a designee will review the artifacts and determine if the package has met all of the requirements, or ask for additional artifacts for any practice or process that does not appear to have sufficient documentation.

After the set of artifacts has been approved, and an individual or a group has been assigned long-term maintenance responsibilities, the package will continue to be eligible for release as a Trilinos package, provided the level of practice and process rigor is maintained and long-term maintenance responsibilities are assigned to someone. Packages that have not met these requirements and are not in the process of doing so should not be included in future Trilinos releases.

5. Exceptional Cases

Although most of the work in Trilinos falls under the above phases, there are some exceptions. We discuss two of these here.

5.1. Isolated Lower Phase Work

Often a package that is in Phase 2 (or Phase 3) will have a small subproject that is research oriented

and is most effectively developed as a Phase 1 (or Phase 2) effort. We allow this exception as long as the subproject work is clearly isolated from the main package source and is reasonably guaranteed to not destabilize the building or execution of the core package capabilities. Isolation can be achieved by one or more of several techniques:

1) Subdirectories: All lower phase software is contained in specially-designated subdirectories and is only activated by special compilation procedures.
2) Interface adapters: Lower phase software is self-contained in new class files and accessed via polymorphism of abstract interface mechanisms or similar techniques that are not necessary for basic operation.
3) Conditional compilation directives: Code that is lower phase is conditionally compiled by define statements. By default, this code should not be compiled. Although acceptable, conditional compilation is discouraged in general, unless the lower phase effort is being done in a special branch of the software repository.

5.2. Externally-developed Packages

Trilinos provides several mechanisms for independently-developed software packages to be used in combination with Trilinos. By default such packages are considered to be in Phase 1, and must go through the promotional events described here to reach a higher level phase. Note that this policy does not affect libraries such as BLAS[7], which are used by Trilinos to provide implementation of interfaces but are not considered Trilinos packages in their own right. These third-party libraries are certified as part of the package test suite for a package that uses them.

6. Conclusions and Future Work

In order to support a broad range of software engineering efforts within the Trilinos Project, we have developed a software lifecycle meta-model. In our study of existing literature, our software environment is not entirely unique. However, our desire to provide an environment that supports development from the inception of high-risk, high-payoff mathematical software to eventual production-quality tools is unusual.

The Trilinos Software Lifecycle Model is a 3-phase meta-model, supporting the spectrum of needs from the early research-oriented phase of leading edge mathematical software to the production quality required for reliable modeling and simulation software. By providing these three phases we allow new packages to evolve quickly, or fail if ideas do not bear fruit, and yet provide motivation and direction for package teams to increase the rigor of their practices by identifying promotional events to reach the next phase.

Because this 3-phase model is newly defined, we will likely need to adjust its description. In particular, although we have had many packages advance from Phase 1 to Phase 2, no package has been elevated to Phase 3 because all Trilinos packages are still under active development using agile-like practices and processes. As a result, future work includes use of the promotional events from Phase 2 to 3, and the practices of Phase 3.

7. References

[1] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, "An Overview of Trilinos," *ACM Transactions on Mathematical Software,* vol. 31, pp. 397-423, 2005.

[2] M. A. Heroux, "Trilinos Home Page." http://trilinos.sandia.gov, 2004.

[3] M. A. Heroux, J. M. Willenbring, and R. Heaphy, "Trilinos Developers Guide Part II: ASCI Software Quality Engineering Practices Version 1.0," Sandia National Laboratories 2003.

[4] R. R. D. Edward A. Boucheron, H. Carter Edwards, Molly A. Ellis, Christi A. Forsythe, Robert Heaphy, Ann L. Hodges, Constantine Pavlakos, Joseph R. Schofield, Judy E. Sturtevant and C. Michael Williamson, "Sandia National Laboratories Advanced Simulation and Computing (ASC) Software Quality Plan Part 2: Mappings for the ASC Software Quality Engineering Practices Version 1.0," Sandia National Laboratories, Albuquerque, NM SAND2004-6601, January 2005 2005.

[5] K. B. w. C. Andres, *Extreme Programming Explained*, Second ed. Boston: Addison-Wesley, 2005.

[6] G. B. Ivar Jacobson, James Rumbaugh, James Rumbaugh, Grady Booch, *The Unified Software Development Process*. Boston: Addison-Wesley, 1999.

[7] S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley, "An Updated Set of Basic Linear Algebra Subprograms (BLAS)," *ACM Transactions on Mathematical Software,* vol. 28, pp. 135-151, 2002-06 2002.