Software Life-cycle and Integration Issues for CS&E R&D Software and Experiences from Trilinos (Part II, Integration Issues)

Roscoe A. Bartlett
Department of Optimization & Uncertainty Estimation
Trilinos Software Engineering Technologies and Integration Lead

Sandia National Laboratories

SIAM Parallel Processing 2010, 2/24/2010
Vision for a Confederation of CS&E Software
Overview of CS&E Software Engineering Challenge

• Progress in Computational Science and Engineering (CS&E) is occurring due to greater numbers of more complex algorithms and methods
  – **Discretization**: a) meshing, b) advanced discretizations, c) adaptively, …
  – **Parallelization**: a) parallel support, b) load balancing, …
  – **General numerics**: a) automatic differentiation, …
  – **Solvers**: a) linear-algebra, b) linear solvers, c) preconditioners, d) nonlinear solvers, e) time integration, …
  – **Analysis capabilities**: a) error-estimation, b) stability analysis and bifurcation, c) optimization, d) UQ, …
  – **New architectures**: a) multi-core, b) GPUs, …
  – **Visualization**
  – …

• Each technology requires specialized PhD-level expertise
• Almost all technologies need to be integrated into single applications
• Set of algorithms/software is too large for any single organization to create
• Too large to be developed under single blanket of Continuous Integration (CI)

Software Engineering and Software Integration are key bottlenecks for CS&E to have the fullest impact!
The Vision: A Confederation of CS&E Codes?

- Develop a confederation of trusted, high-quality, reusable, compatible, software packages/components including capabilities for:
  - **Discretization**: a) meshing, b) advanced discretizations, c) adaptively, …
  - **Parallelization**: a) parallel support, b) load balancing, …
  - **General numerics**: a) automatic differentiation, …
  - **Solvers**: a) linear-algebra, b) linear solvers, c) preconditioners, d) nonlinear solvers, e) time integration, …
  - **Analysis capabilities**: a) error-estimation, b) stability analysis and bifurcation, c) optimization, d) UQ, …
  - **New architectures**: a) multi-core, b) GPUs, …
  - **Visualization**
  - …

---

Trilinos itself is a smaller example of this!
Requirements/Challenges for Confederation of CS&E Codes

- **Software quality and usability**
  => Design, testing, collaborative development

- **Building the software in a consistent way and linking**
  => Common build approach?

- **Reusability and interoperability of software components**
  => Incremental Agile design

- **Documentation, tutorials, user comprehension**
  => SE education, better documentation and examples

- **Critical new functionality development**
  => Closer development and integration models

- **Upgrading compatible versions of software**
  => Frequent fixed-time releases

- **Safe upgrades of software**
  => Regulated backward compatibility, software quality

- **Long term maintenance and support**
  => Stable organizations, stable projects, stable staff

- **Self-sustaining software** (clean design, clean implementation, well tested with unit tests and system verification tests)  => Anyone can maintain it!

The Trilinos is taking (baby) steps to address all of these issues at some level.
Software Integration Strategies
Sophisticated CS&E applications
- Discretized PDEs (SIERRA, Alegra, Aleph, Charon)
- Circuit network models (Xyce)
- Other types of calculations (Titan/VTK, Tramonto)

(Massively) parallel MPI (Gordon Bell Winners)
Almost entirely developed by non-software people
Wide range of research to production (i.e. from Aleph to SIERRA)

TPL: Third Party Lib
- Provides functionality to multiple APPs
- The “Supplier” to the APP

APP: Application
- Delivers end user functionality
- The “Customer” of the TPL
Standard Software Integration Approaches

• Continuous Integration (CI)
  – Code is expected to build and the tests are expected to run
  – Maintained through synchronous or asynchronous CI
  – Requires high levels of cooperation and communication
  – Requires code to (re)build fast and tests to run fast

• Customer/Supplier Relationships
  – Combined code too large to build under single CI system
  – Organizations can not cooperate close enough
  – Protect APP for future TPL updates through development of Acceptance Test Suite
  – May not work as well for many CS&E codes
  – Not as well suited for closer collaborations
Special Challenges with CS&E Software

- CS&E heavily relies on fast floating-point computations
  - Output from program varies between platforms and even with different compiler options!
  - How to you keep tests working on different platforms?

- CS&E involves complex nonlinear models
  - Examples: ill conditioning, multiple solutions, bifurcations, non-convexities ...

These issues conspire together to make testing and maintaining CS&E software on multiple platforms very difficult!

Consequences:
- A new test status: The diffing test!
  - Code runs to completion but some error tolerance was exceeded
  - Many CS&E practitioners convince themselves that a “diff” is not as bad as a “fail”!
- Changes to a numerical algorithm that improve performance in every measure can cause numerous tests to ‘diff’ or even ‘fail’!
- Upgrades of a TPL can break an APP even if no real defects have been introduced!
• Transition from TPL X to TPL X+1 can be difficult and open ended
• Large batches of changes between integrations
• Greater risk of experiencing real regressions
• Upgrades may need to be completely abandoned in extreme cases
• However, this is satisfactory for many APP+TPL efforts!
APP + TPL Release and Dev Daily Integration

- APP Dev Developers only build/test against TPL Release
- TPL (Trilinos) Dev Developers work independent from APP
- Keep APP Dev and TPL Dev up to date! => Supported by TPL backward Compatibility!
- Use of staggered releases of TPL and APP
- APP + TPL Dev Co-Developers drive new capabilities
- Difficult for APP to depend too much on TPL
- Does not support tighter levels of integration and collaboration
- APP developers can break “New” a lot when refactoring
- However, this is satisfactory for many APP+TPL efforts!
APP + TPL Release and Dev Daily Integration

- Testing: APP Dev + TPL Dev
- Testing: APP Dev + TPL X

Charon + Trilinos Integration!
Alegra + Trilinos Integration!
Xyce + Trilinos Integration!

- All changes are tested in small batches
- Low probability of experiencing a regression
- Extra computing resources to test against 2 (3) versions of TPL
- Some difficulty flagging regressions of APP + TPL Dev
- APP developers often break APP + TPL Dev when refactoring
- Difficult for APP to rely on TPL too much
- Hard to verify TPL for APP before APP release
- However, this is satisfactory for many APP+TPL efforts!
APP + TPL Almost Continuous Integration: Overview

APP Owned

APP Dev
Nightly Testing
APP Dev + TPL Dev

APP Dev + TPL Dev

APP Owned

Main APP VC Repository (Dev)

APP-owned TPL VC Repository (Dev-)

APP Pre-Checkin Test Suite

APP Regression Test Suite

TPL Owned

TPL Dev
Nightly Testing

TPL Dev

Main TPL VC Repository (Dev)

TPL Pre-Checkin Test Suite

TPL Regression Test Suite

APP Dev + TPL Dev Co-Developers

TPL Dev Developers

APP Dev Developers
APP + TPL Almost Continuous Integration: Releases

**TPL Head (Dev)**

**SIERRA + Trilinos Integration!**

**APP Head (Dev)**

Nightly Testing: APP Dev + TPL Dev (pre-checkin tests only, TPL Dev- checkin)
Nightly Testing: APP Dev + TPL Dev- (complete test suites)
Supported with asynchronous continuous integration testing of APP Dev + TPL Dev

- All changes are tested in small batches
- Low probability of experiencing a regression between major releases
- Less computing resources for detailed nightly testing (only one TPL version)
- All tested regressions are flagged in less than 24 hours
- Allows code to flow freely between the APP and TPL
- Supports rapid development of new capabilities from top to bottom
- All code checked out by APP Dev developers has passed pre-checkin build/test
- More complex processes (i.e. requires some tools?)
- APP Dev developers spend more time spent recompiling TPL code
- Recommended for projects requiring high levels of integration & collaboration!
APP + TPL Integration: Different Collaboration Models

• Absorb sources for TPL and never upgrade
  • Just a code seeding strategy and not an integration strategy

• APP + TPL Release with Punctuated TPL Upgrades
  – Little to no testing of APP + TPL Dev in between versions

• APP + TPL Release and Dev Daily Integration
  – APP developers work against TPL Release
  – APP + TPL team(s) build against TPL Dev
  – Nightly and CI testing done for both APP + TPL Release and Dev
  – Must handled staggered releases of TPL and APP

• APP + TPL Almost Continuous Integration
  – APP developers work directly against TPL Dev checked out every day
  – Releases best handled as combined releases of APP and TPL
Selecting an Integration Model for CS&E Software

• Each of these different integration models will be appropriate for a particular APP+TPL situation.

• The particular integration model can be switched during the life-cycles of the APP and TPL depending on several factors:
  – Level of dependence on TPL?
  – Level of duplication of functionality in TPL with other external packages?
  – Level of sophistication of TPL?
  – Ease or difficulty of independent verification of TPL?
  – Level of active development of TPL?
  – Need for new functionality and upgrades of TPL?
  – Interdependence of TPL on other external software packages?
  – Level of quality needed for TPL?
  – Level of Software Quality Engineering used to produce TPL?
  – Release schedule for TPL?
  – Level of relationship and pull with the developers of TPL?
  – Stability of the organization that develops and supports TPL?
  – Usage of TPL by other related sister codes?
  – ...
Maintenance of APP + TPL Integration

- **APP + TPL Monitor:**
  - Member of the APP development team
  - Has good familiarity with the TPLs
  - Performs first-round triage (APP or TPL?)
  - Forwards issues to APP or TPL Reps
  - Ultimate responsibility to make sure issues are resolved

- **APP Representative:**
  - Member of the APP development team
  - Second-round triage of APP issues
  - Forwards hard APP issues to APP developers

- **TPL Representative:**
  - Member of the TPL development team
  - Has some familiarity with the APPs
  - Second-round triage for TPL issues
  - Forwards hard TPL issues to TPL developers

- **General principles:**
  - Roles of authority and accountability (Ordained by management)
  - At least two people serve in each role
  - Rotate people in roles
Summary
Progress in Computational Science and Engineering (CS&E) is occurring due to greater numbers of more complex algorithms and methods

- **Discretization**: a) meshing, b) advanced discretizations, c) adaptively, ...
- **Parallelization**: a) parallel support, b) load balancing, ...
- **General numerics**: a) automatic differentiation, ...
- **Solvers**: a) linear-algebra, b) linear solvers, c) preconditioners, d) nonlinear solvers, e) time integration, ...
- **Analysis capabilities**: a) error-estimation, b) stability analysis and bifurcation, c) optimization, d) UQ, ...
- **New architectures**: a) multi-core, b) GPUs, ...
- **Visualization**
- ...

- Each technology requires specialized PhD-level expertise
- Almost all technologies need to be integrated into single applications
- Set of algorithms/software is too large for any single organization to create
- Too large to be developed under single blanket of Continuous Integration (CI)

Software Engineering and Software Integration are key bottlenecks for CS&E to have the fullest impact!
Summary of CS&E Software Integration Models

• Nightly building and testing of the development versions of the application and TPLs:
  – results in better production capabilities and better research,
  – brings TPL developers and APP developers closer together allowing for a better exchange of ideas and concerns,
  – refocuses TPL developers on customer efforts,
  – helps drive continued research-quality TPL development, and
  – reduces barriers for new TPL algorithms to have impact on production applications.

• Integration Models:
  – APP + TPL Release with Punctuated TPL Upgrades
    • Little to no testing of APP + TPL Dev in between TPL releases
  – APP + TPL Release and Dev Daily Integration
    • Daily Integration testing done for both APP + TPL Release and Dev
    • Staggered releases of TPL and APP
  – APP + TPL Almost Continuous Integration
    • APP Dev + TPL Dev developers update both APP-owned and main TPL repositories
    • Nightly testing of APP Dev + TPL Dev automatically updates APP-owned TPL Dev- VC Repository
    • Releases best handled as combined releases of APP and TPL
    • TPL Dev- checkins can be dialed back approaching TPL Release and Dev Integration!
Requirements/Challenges for Confederation of CS&E Codes

• Software quality and usability
  => Design, testing, collaborative development
• Building the software in a consistent way and linking
  => Common build approach?
• Reusability and interoperability of software components
  => Incremental Agile design
• Documentation, tutorials, user comprehension
  => SE education, better documentation and examples
• Critical new functionality development
  => Closer development and integration models
• Upgrading compatible versions of software
  => Frequent fixed-time releases
• Safe upgrades of software
  => Regulated backward compatibility, software quality
• Long term maintenance and support
  => Stable organizations, stable projects, stable staff
• Self-sustaining software (clean design, well tested with unit tests and system verification tests)  => Anyone can maintain it!

Possible topics for Round Table Discussion at 6:00 PM