SAND2007-7236C

Daily Integration and Testing of the Development Versions of Applications and Trilinos

A stronger foundation for enhanced collaboration in application and algorithm research and development

Roscoe A. Bartlett
Department of Optimization & Uncertainty Estimation

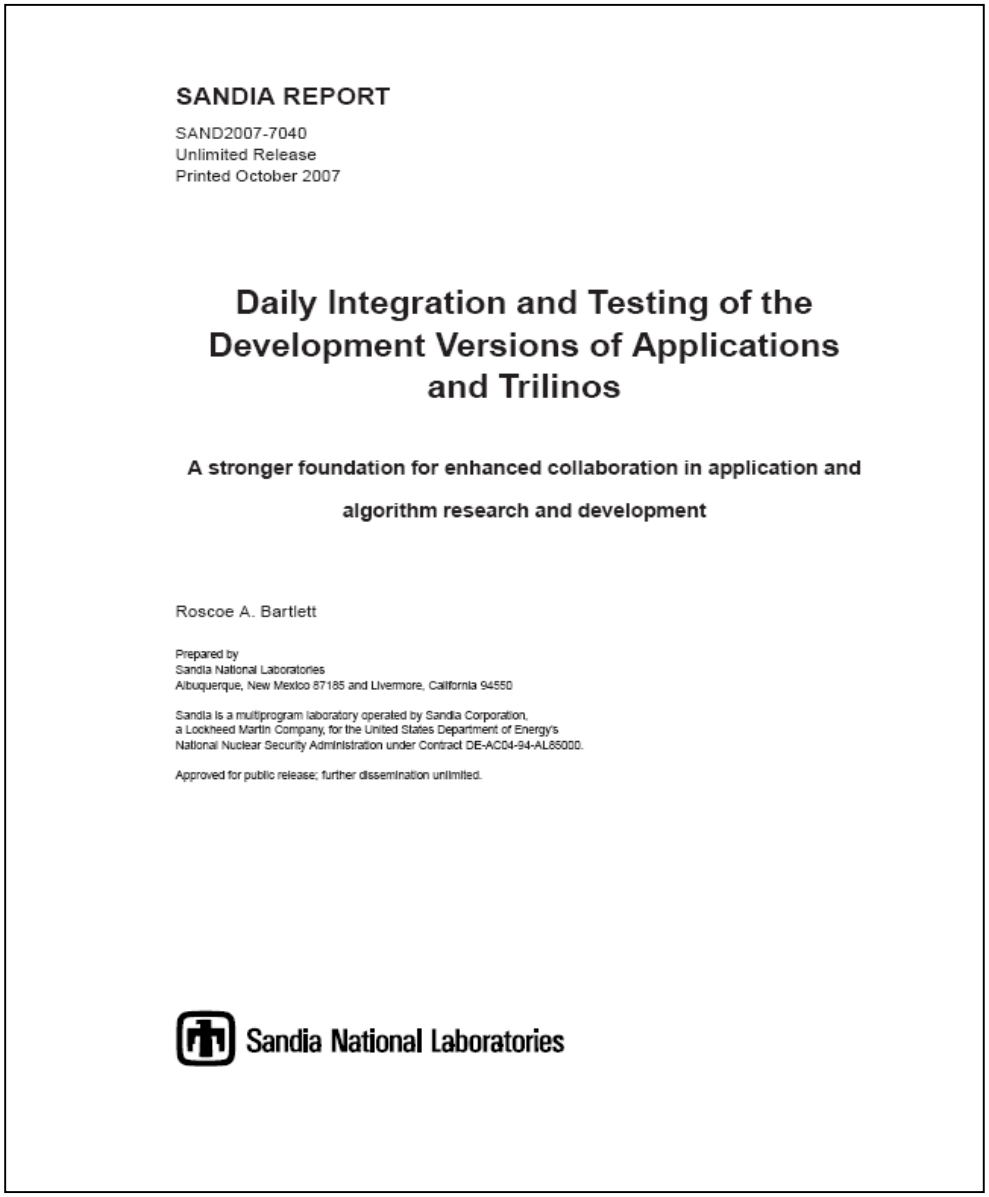
Sandia National Laboratories

Software Engineering Seminar Series, October 30th, 2007

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.

Sandia National Laboratories

APP + Trilinos Dev SAND Report



SANDIA REPORT

SAND2007-7040
Unlimited Release
Printed October 2007

Daily Integration and Testing of the Development Versions of Applications and Trilinos

A stronger foundation for enhanced collaboration in application and algorithm research and development

Roscoe A. Bartlett

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's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

http://software.sandia.gov/SESS/common/AppPlusTrilinosDev.pdf

Sandia National Laboratories

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Overview of Current APP + Trilinos Interactions

- Many active research collaborations have existed and continue to exist between applications (APPs) and Trilinos where APP + Trilinos Dev is developed throughout year and drives algorithm and APP development
- Currently, Trilinos has a major release once a year (typically in September, at the end of each Fiscal Year)
  - Some minor releases go out from time to time (e.g. Trilinos 7.1.x for ML)
  - Several minor minor releases (bug fixes) go out frequently (e.g. 7.0.9)
- Potential issues
  - Porting of some Trilinos packages to some customer platforms is only done right before a major release => Portability problems
  - May be very difficult to update the APP for new Trilinos release (e.g. porting problems, failing and diffing tests, etc.)
  - As much as year or more between introduction of new Trilinos capability and impact on end customer (through an APP release)
  - Post delivery maintenance after the introduction of a new capability

Obstacles to Algorithms Impacting Applications

- Technical Challenges
  - Features needed by algorithm from APP are difficult to implement
    e.g. Accurate parameter derivatives, in-core response functions
  - Algorithm is unproven on APP problem area
  - Inability of algorithm to scale to APP sizes (# DOFs and # processes)
  - Algorithm too complex to easily implement by APP developers
    => Have algorithm experts implement the algorithms =>Trilinos!
  - Others …
- Difficulty of collaboration between algorithm and APP developers
  - Difficult to access shared platforms
  - Difficult to build Dev versions algorithm and APP codes together
    - Software configuration management problems
    - Development versions of APP and algorithm codes incompatible

  => Even smallest overhead can kill a collaboration

Obstacles to Algorithms Impacting Applications

- Examples of missed opportunities in FY07
  - Aria + Rythmos
    - Pat Notz (1514) had some funding to put Rythmos into Aria
    - There where some issue in the ModelEvaluator software
    - Could not easily access Dev version of Trilinos to address issues!
  - Alegra + Thyra/Stratimikos
    - Tom Brunner (1641) wanted to experiment with Thyra/Stratimikos to build some physics-based preconditioners

      PLUG: Thyra talk 3:30 PM Tuesday 11/6 at Trilinos User Group (TUG) Meeting in CSRI/90!

      http://trilinos.sandia.gov/events/trilinos_user_group_2007
    - Using released version of Trilinos and must port to Purple => Porting problems
    - Could not easily access Dev version of Trilinos to address issues!

Future of APP + Trilinos Interactions?

- Many applications are reluctant to take on direct Trilinos dependences
  - Example: APP will not allow use of Teuchos::RCP at application level
- Trilinos capability set is expanding:
  - Automatic differentiation (Sacado)
    - A fundamental type of dependency!
  - PDE discretization support (Intrepid)
  - Mesh support (phdMesh)
  - Finite element assembly (FEI)
  - …
- Will applications take on these new dependences in a significant way?
  - Can algorithm development in Trilinos lead to "Transformational" changes in APPs when treaded as a Third Party Library (TPL)?
- My assertion: For applications to take on more serious Trilinos dependences they must have more direct interaction with and control of these services!

Sandia National Laboratories

ASC FY07 Level-2 Vertical Integration Milestone (Targets)

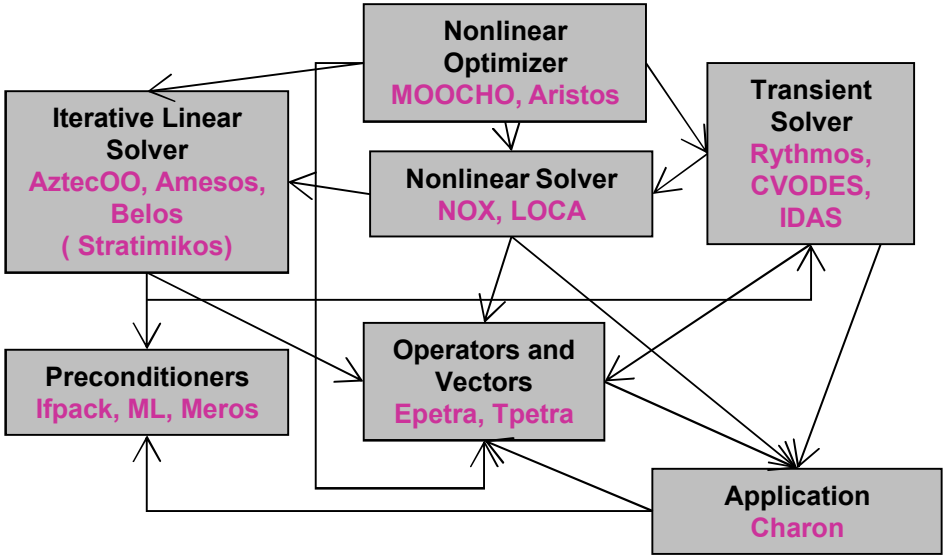
Steady-State Simlaton-Constrained Optimization:

Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that:
minimizes $g(x,p)$
such that $f(x,p) = 0$

x: state variables
p: optimization parameters

Transient Simulation-Constrained Optimization:

Find $x(t) \in \mathbf{R}^n$ in $t \in [0,T]$ and $p \in \mathbf{R}^m$ that:
minimizes $\int_0^T g(x(t),p)$
such that $\dot{x} = f(x(t),p,t) = 0$, on $t \in [0,T]$
where $x(0) = x_0$



Targeted Demonstration Problems:

- Steady-state and transient model calibration against experimental data (part of QASPR project) modeled with Charon using parameter estimation optimization using Rythmos, MOOCHO, …

Goal: Demonstrate fully vertical integration from linear algebra all the way through to optimization

Approach: Create standard interfaces and implementations to break N-to-N coupling using Thyra

Sandia National Laboratories

ASC FY07 Level-2 Vertical Integration Milestone (Achieved)

- **Goal:** Vertically integrate newly developed advanced numerical solver algorithms in Trilinos to build new predictive capabilities
  - **Impact:** Achieved vertical integration of more than 10 Trilinos algorithm packages from parallel linear algebra data structures to transient sensitivities and simulation-constrained optimization!
- **Goal:** Demonstrate new vertically integrated solver algorithms on relevant production applications
  - **Impact:** Solved steady-state parameter estimation problems and transient sensitivities on QASPR-related semiconductor devices => New capabilities!
- **Goal:** Deliver new vertically integrated solvers to ASC customers
  - **Impact:** Release of new packages in Trilinos 8.0!
- **Added Goal:** Explore enhanced models of collaboration between production application developers and algorithm researchers.
  - **Impact:** Closer collaboration between application and algorithm developers yielding better solvers and better applications

Charon + Trilinos Dev Daily Integration &Testing

- Motivation for nightly building and testing of Charon + Trilinos Dev
  - Avoid backslides in Vertical Integration Milestone capabilities
    - Before nightly testing, backslides of capability happened!
  - Proof of Milestone capabilities
  - Preservation of milestone problems

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck (XP)
Mike Beedle
Arie van Bennekum
Alistair Cockburn (Crystal Clear)
Ward Cunningham
Martin Fowler (UML, Refactoring)
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

http://agilemanifesto.org

Important Aspects of Agile (and other) Software Methods

- Frequent Integration of the Software Product is Critical

  "Whatever integration strategy you select, a good approach to integrating the software in the "**daily** build and smoke test"."

  Steve McConnell, *Code Complete: Second Edition*

- Close Customer Involvement and Interaction is Key

  "In order for a project to be agile, there must be **significant and frequent** interaction between the customers, developers, and stakeholders"

  Robert C. Martin, *Agile Software Development*

- Frequent Delivery of Capability to Customers is a Top Priority

  "The single most important property of any project, large or small, agile or not, is that of delivering running, tested code to real users **every few months**. The advantages are so numerous that it is astonishing that any team doesn't do this."

  Alistair Cockburn, *Crystal Clear*

Questions: Where does Trilinos and APP fit in? Who is the "customer", "developer", "stakeholder"?

Where does Trilinos Fit It?

- The term “Agile” has been kicked around for research-driven software
  - All “Agile” methods involve Frequent Integration and Frequent Delivery
  - What does this mean for Trilinos?
- What is Trilinos with respect to “the product”, “the customer”, software integration, and software delivery?
  - Is Trilinos an integrated project just within itself?

    =>Just integrate Trilinos with itself and we are Agile?
  - Is Trilinos critical enabling technology that is used in real products?

    =>Trilinos needs to frequently integrate with APPs to be Agile?
  - Do APP requirements continue to drive Trilinos development?

    =>Trilinos needs to frequently integrate to let APPs drive Trilinos?
- I would argue that for Trilinos to be “Agile”, we have to frequently integrate with at least our immediate APP customers (even when active research collaborations are not taking place)
- What about frequent delivery? => I will not directly address this here!

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Principles for APP + Trilinos Dev Daily Integration & Testing

- Support active collaboration between APP and Trilinos developers
- Guard against destabilizing the separate development of APP and Trilinos
- Minimize unnecessary interactions between APP and Trilinos Developers
- Streamline communication between APP and Trilinos developers when such communication is needed
- Accountability for keeping APP + Trilinos Dev working

Outline for APP + Trilinos Dev Integration & Testing

- APP developers work mainly with APP + Trilinos Release

  => Minimize unnecessary interaction with Trilinos developers

- Trilinos developers work mainly with Trilinos Dev

  => Minimize unnecessary interaction with APP developers

- Hide APP + Trilinos Dev “research” work in APP behind ifdefs

  => Keep both versions of the code close together (no branches)

- Segregate “production” and “research” tests

  => Differentiate between APP and Trilinos defects

- Perform nightly building and testing of APP + Trilinos Release and APP + Trilinos Dev

- Release APP + Trilinos together or staged

- Continue APP + Trilinos Dev nightly building and testing after APP upgrades to new Trilinos release

Outline for APP + Trilinos Dev Integration & Testing

- Perform nightly building and testing of APP + Trilinos Release/Dev
  - APP + Trilinos Release tested against “production” tests
    - Send “production” failures only to APP developers
  - APP + Trilinos Dev tested against “research” and “production” tests
    - Only send “production” failures that did not also fail in APP + Trilinos Release to Trilinos developers
    - All “research” failures go to appropriate Trilinos (or APP) developers

Outline for APP + Trilinos Dev Integration & Testing

- Release APP + Trilinos together or staged
  - Combined tagging and release of APP + Trilinos
    - Requires instant releasability of Trilinos
    - Advantage: Most up to date algorithms to customers
    - Disadvantage: Supporting multiple Trilinos releases
  - Staged releases of Trilinos and APP
    - Two approaches to handle Trilinos release
      - a) APP updates to new Trilinos Release right away
        - Build & test against i) Trilinos Release and ii)Trilinos Dev
      - b) APP delays update to new Trilinos release
        - Build & test against i) old Trilinos release, ii) current Trilinos release, and iii) Trilinos Dev
    - Next APP release is performed against latest Trilinos Release
    - Advantage: Fewer, more predictable Trilinos releases
    - Disadvantage: More delay in getting new Trilinos capabilities to end customers

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Research & Production Advantages for APP + Trilinos Dev

- Research Advantages
  - Reduces overhead for initial algorithm integration
  - Improves chances that new algorithms will have impact
  - Preserves interesting/challenging problems
- Production Advantages
  - Expands testing for Trilinos
  - Enables better scalability testing for Trilinos
  - Reduces time to detection of defects
  - Reduces release time and effort
  - Allows for more aggressive refactorings and code improvements
  - Better address customer needs
  - Reduces all kinds of risk

Research Advantages for APP + Trilinos Dev Integration

- Reduces overhead for initial algorithm integration

  => Development versions build right away and all tests pass!

- Improves chances that new algorithms will have impact

  => Lower overhead!

- Preserves interesting/challenging problems
  - Not such a big deal for linear solvers and preconditioners
  - Very big issue for higher level algorithms like invasive sensitivities and optimization
  - Provides basis of comparison for showing “progress” in our algorithms
  - Provides a spring board for related efforts

Production Advantages for APP + Trilinos Dev Integration (1)

- Expands testing for Trilinos
  - Installation testing for Trilinos
  - Catches errors that Trilinos test suite does not (happened several times with Charon + Trilinos Dev)
- Enables better scalability testing for Trilinos
  - APP provide scalable test problems that can test Trilinos scalability
  - Teuchos timers can help
- Reduces time to detection of defects
  - Expended test => earlier detection of critical defects
  - Cost to fix a defect increases with time between when defect is introduced to when it is first detected! (*Code Complete: 2nd edition*)
  - Shows the 24 hour period where defect is introduced!

Production Advantages for APP + Trilinos Dev Integration (2)

- Reduces release time and effort (THIS IS HUGE)
  - Insures APP + Trilinos Release will build right away
  - Insures APPs test suite will pass right away
  - Makes release cost and schedule much shorter more predictable
- Allows for more aggressive refactorings and code improvements (THIS IS HUGE FOR ME)
  - Trilinos developers can directly update APPs for changes in Trilinos interfaces or behavior
  - APP developers can request more aggressive refactorings to suite APP needs and can even modify Trilinos code themselves
  - Changes to the interface or behavior of Trilinos code can be tested against APPs right away
  - This is the #1 issue for post delivery maintenance!
  - For me, this is one of the biggest motivations for daily integration and testing of APP + Trilinos Dev! => Truly Agile development!

Production Advantages for APP + Trilinos Dev Integration (3)

- Better address customer needs
  - Brings Trilinos and APP developers closer together
  - Lets APP developers drive targeted Trilinos algorithm development
- Reduces all kinds of risk
  - Reduces risk that APP + Trilinos will regress
  - Reduces risk of slipping release deadlines
  - Reduces risk of overtime needed to repair broken capability
  - Reduces risk that Trilinos algorithm development will have limited impact
  - …

Potential Disadvantages for APP + Trilinos Dev Integration

- Will slow down day-to-day development to varying degrees
  - Guaranteed to slow down day-to-day work some
  - However, experience of other suggest that overall development, release, and support effort will actually decrease!
- Will require better, more coordinated management practices
  - Access to each others code repositories
  - Access to common computing environments
- Will impose greater responsibility to meet customer needs
  - Algorithm developers will have greater responsibility to meet real customer needs!
  - However, there is always a place for more fundamental (i.e. less a applied) algorithm research.
- Could increase overall development effort
  - It could, but again experience of others suggest just the opposite

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Outline of Suggested Practices for APP + Trilinos Dev

- Separate ``production'' and ``research'' tests
- Refactor APP code to isolate and separate ifdefed code
- Maintain a dedicated machine for building and testing APP + Trilinos Dev
- Appoint a dedicated APP + Trilinos Representative
- Provide easy access for any Trilinos or APP developer to build, test, and develop APP + Trilinos Dev
- Fix failed builds of APP + Trilinos Dev ASAP
- Address failing ``research'' and ``production'' tests on a schedule appropriate for the APP + Trilinos collaboration
- Archive test results for sufficiently long periods of time
- Transition ``research'' to ``production'' appropriately after each Trilinos release
- Perform APP + Trilinos Release and APP + Trilinos Dev nightly testing on the same set of platforms

Suggested Practices for APP + Trilinos Dev Integration (1)

- Separate ``production'' and ``research'' tests
  - Helps differentiate between APP and Trilinos defects
  - Helps avoid unnecessary communication between APP and Trilinos developers
- Refactor APP code to isolate and separate ifdefed code
  - Makes code development and maintenance easier and safer
  - e.g. Factor out code into different functions
- Maintain a dedicated machine for building and testing APP + Trilinos Dev
  - Main platform for nightly building and testing of APP + Trilinos Dev
  - Provides low overhead “sandbox” for APP and Trilinos developers to try out new things
  - Only require SRN (or SON in some case) access!!!!

Suggested Practices for APP + Trilinos Dev Integration (2)

- Appoint a dedicated APP + Trilinos Representative (VERY IMPORTANT)
  - Point of contact to make sure failing builds and tests are addressed
  - Facilitates communication between APP and Trilinos developers
  - Maintains the dedicated APP + Trilinos Dev machine
  - Ideally already an active APP and Trilinos developer
  - Goal => Less than 0.15 FTEs of effort!
- Provide easy access for any Trilinos or APP developer to build, test, and develop APP + Trilinos Dev
  - Getting a new APP or Trilinos devleoper access to a private build of APP + Trilinos Dev should take less than 10 minutes!
  - Greatly helped by having a dedicated machine with spare cycles
- Fix failed builds of APP + Trilinos Dev ASAP (VERY IMPORTANT)
  - A failed build means no test results at all!
  - Keeping APP + Trilinos Dev should be pretty easy

Suggested Practices for APP + Trilinos Dev Integration (3)

- Address failing ``research'' and ``production'' tests on a schedule appropriate for the APP + Trilinos collaboration
  - Two extremes:
    - a) All test failures are given highest priority and fixed ASAP
    - b) APP + Trilinos Dev is just kept building and no tests are fixed until just before a release
  - A more moderate approach:
    - Take 10 minutes to research failing test and send off e-mails or file bug reports to guilty parties
    - Address less critical failing tests every Thursday?
  - Urgency in addressing failing tests depends on nature of APP and current APP + Trilinos collaborations
  - Knowing the 24 hour period where a test first fails is critical

Suggested Practices for APP + Trilinos Dev Integration (4)

- Archive test results for sufficiently long periods of time
  - Must be able to compare test output for consecutive days when test went from passing to failing
  - Keep all output files for 24 hours before next test run
  - Archive only smaller files that are needed to examine algorithm behavior and diagnose test failures
  - Provide access to test results through a web server
- Transition ``research'' to ``production'' appropriately after each Trilinos release
  - Solid “research” tests become “production” tests after a release which puts responsibility on APP developers not to break them
- Perform APP + Trilinos Release and APP + Trilinos Dev nightly testing on the same set of platforms
  - Avoid porting and rounding issues from building and running APP + Trilinos Release and APP + Trilinos Dev on different platforms
  - Helps avoid unnecessary communication/interaction between APP and Trilinos developers

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

Experience from Charon + Trilinos Dev from Milestone Work

- Details of Charon + Trilinos Dev Daily Integration and Testing
  - Shell (sh) scripts to created to:
    - Check out Dev versions of Charon, Charon_TPLs and Trilinos
    - Build Charon + Trilinos Release/Dev
    - Run test suite
    - Analyze results
    - Archive test results
    - Send out e-mail notifications
  - For most of milestone, nightly building and testing of:
    - Charon + Trilinos Dev opt (from updated sources, all tests)
    - Charon + Trilinos Dev dbg (from scratch, only milestone tests)
  - All builds, tests etc. run on my own personal Linux workstation (64 bit, AMD, Fudora Linux 4.0, GCC 3.4.6, …)
  - Results archived and accessible through web server on test (i.e. my) machine and linked to in notification e-mails

Experience from Charon + Trilinos Dev from Milestone Work

Example notification e-mail for a failing test

Charon tests failed: tridev+dbg: passed=33, notpassed=1

Build: 2D_64BITgnu3_dbg_hessian_tridev

Summary: 33 pass, 0 timeout, 0 diff, 1 fail, 0 notrun, 0 notdone

The build of Charon seemed to work but at least one of the tests did not pass (see below).

X defects-moocho Exit fail(1) 2s
Semiconductor/2D/nodal/sensitivity/defects/9_elem_diode/defects-moocho.np=1

See the test output in http://gabriel.sandia.gov/charon-tests/2007-10-24-00-00/2D_64BITgnu3_dbg_hessian_tridev

- A few clicks of the mouse and you can view test results and start to diagnose the problem

Experience from Charon + Trilinos Dev from Milestone Work

- Timeline for Charon + Trilinos Dev Daily Integration and Testing
  - ???: Roger Pawlowski gets Charon building against Trilinos Dev
  - 10/30/06: First Charon+Trilinos Dev milestone-related test added
  - 12/2/06 - 1/7/07: Did not build or test Charon + Trilinos Dev
  - 1/7/07: Rebuild Charon + Trilinos Dev and a test failed!
  - 1/27/07: Deployed initial nightly building & testing of Charon + Trilinos Dev
  - 7/9/07: Branch for Trilinos 8.0
    - We did not build against Trilinos 8.0 (NOT GOOD)
  - 7/24/07: Charon updated to build only against Charon + Trilinos 8.0
    - We did not build against Trilinos Dev (NOT GOOD)
  - 9/21/07: Nightly testing updated to build against Trilinos 7.0, 8.0, and Dev
    - **This is the way to do it!**
  - 10/18/07: Nevada refactoring committed to Charon => Charon + Trilinos Dev goes down!
  - 10/25/07: Nightly testing of Charon + Trilinos 8.0 & Dev back up (Not 7.0!)
  - ???: Charon updated to Trilinos 8.0, drops support for Trilinos 7.0

Experience from Charon + Trilinos Dev from Milestone Work

- A few observations Charon + Trilinos Dev Daily Integration and Testing
  - Only a few of weeks total developing and maintaining test scripts
  - Over the course of the milestone, I was able to diagnose most failed builds and failing tests and send out emails in less than 10 minutes
  - All major milestone capabilities are tested every night
  - Charon + Trilinos Dev Daily integration provides a solid foundation for future algorithmic work

Outline

- Background
- Proposed APP + Trilinos Dev Daily Integration and Testing
- Advantages and Disadvantages of APP + Trilinos Dev Daily Integration
- Recommended Practices to Support APP + Trilinos Dev Daily Integration
- Experience from the Vertical Integration Milestone (Charon + Trilinos Dev)
- Wrap Up and Next Steps

APP + Trilinos Dev Checklist

- ❑ Do you have ifdefs in place in APP code to build against Trilinos Dev and against a stable release (or multiple releases) of Trilinos?
- ❑ Have you separated ``production'' and ``research'' tests?
- ❑ Have you appointed an official APP + Trilinos Representative?
- ❑ Have you set up a dedicated machine to do nightly building and testing of APP + Trilinos Dev?
- ❑ Have you provided easy access to APP and Trilinos developers to immediately build a private version of APP + Trilinos Dev?
- ❑ Do you fix failing builds of APP + Trilinos Dev right away, with no exceptions?
- ❑ Do you address failing ``production'' and ``research'' tests with an urgency that is appropriate for the nature of APP and the APP + Trilinos collaboration?
- ❑ Do you archive test results long enough to diagnose failing tests?
- ❑ After each major release of Trilinos, do you upgrade APP to the new release in a timely way?
- ❑ During the transitionary period between Trilinos branch and release do you build APP against the old Trilinos release, the current Trilinos release, and Trilinos Dev?

Conclusions

- Nightly building and testing of the development versions of the application and Trilinos:
  - results in better production capabilities and better research,
  - brings algorithm developers and application developers closer together allowing for a better exchange of ideas and concerns,
  - refocuses Trilinos developers on customer efforts,
  - helps drive continued research-quality algorithm development, and
  - reduces barriers for new algorithms to have impact on production applications.
- Other application projects and scientific support software projects should consider adopting the type of continuous integration that is used with Charon + Trilinos that was developed as part of the ASC Vertical Integration Milestone work.

Next Steps for APP + Trilinos Dev

- Charon + Trilinos Dev Daily Integration and Testing
  - Continuing Nightly building and tested indefinently?
  - Upgrade scripts to use Python?
  - Who will take over as Charon + Trilinos Representative?
  - What new algorithms research will this support?
- Aria/SIERRA + Trilinos Dev Daily Integration and Testing
  - Machine/enviornment being pursued by Russell Hooper (SIERRA + Trilinos Rep.) and Jim Stewart and others
  - Being set up to support Aria/Rythmos ASC Outer Core work and Fuego/ML LDRD work
- Others?
- Continuous integration Server? *Continuous Integration*, Paul Duvall et. al
  - Build and test Trilinos and APP + Trilinos several times a day
  - Avoid checking out broken code
  - …

The End