



2008-7716P

APP + Trilinos Integration

Status, Opportunities, and Challenges

Roscoe A. Bartlett

<http://www.cs.sandia.gov/~rabartl/>

Department of Optimization & Uncertainty Estimation

Sandia National Laboratories

Trilinos User Group Meeting, October 23, 2008



Current Status of APP + Trilinos Integration

- **Charon + Trilinos Integration:**

- ASC FY07 Vertical Integration Milestone
- Automated daily integration testing done against Trilinos 7.0, 8.0, and Dev
- Charon could not upgrade to Trilinos 8.0 last year because Xyce did not upgrade
- Charon will release against Trilinos 7.0 later this quarter!
- John Shadid is building and running Charon against recent snapshots of Trilinos Dev
- Charon was not tested against Trilinos 9.0 and there is no interest from the semiconductor side
- What about Xyce + Charon + Trilinos integration?
- Conclusion: Current Trilinos release process has little impact on Charon

- **Xyce + Trilinos Integration:**

- Ad-hoc manual integrations with Xyce + Trilinos Dev was done prior to branch of Trilinos Release
- Testing of Xyce + Trilinos 9.0 brach was *not* done => Xyce can not build against Trilinos 9.0 and it has consumed several weeks worth of effort!



Current Status of APP + Trilinos Integration

- **Alegra + Trilinos Integration:**

- Automated testing of Alegra + Trilinos Dev was conducted before Trilinos 9.0 branch ... Several problems were resolved before Trilinos 9.0 was branched!
- Alegra switched over to Trilinos 9.0 after branch (but stopped testing against Trilinos Dev)
- New Xyce TPL depends on Trilinos with versioning issues
- Alegra was ready to switch to Trilinos 9.0 by 10/15 but could not because Xyce does not build against Trilinos 9.0!
- Alegra would like to do a Alegra + Xyce + Trilinos Dev daily integration to support their work!

- **Aleph + Trilinos Integration:**

- They take snapshot of Trilinos Dev from time to time and build against that
- Automated testing of Aleph against Trilinos Dev Shapshot
- Aleph would be very interested to use the STK IO capability that might be moved into Trilinos?
- In this case, they would be very interested in doing daily integration with Trilinos Dev ...



Current Status of APP + Trilinos Integration

- Titan/VTK + Trilinos Integration:

- They currently do informal builds against snapshots of Trilinos Dev
- They have already experienced a regression between updated snapshots
- They want to move to automated daily integration testing with Trilinos Dev

- SIERRA + Trilinos Integration:

- Driven by Algorithm Integration Project
 - Embedded algorithms in SIERRA
- SIERRA does *not* use the Trilinos build system, they build Trilinos with BJAM
- Developer environment built constructed with Python scripts (STANA scripts)
- Daily integration testing for all of SIERRA + Trilinos Release and Dev
- Continuous Integration testing done every two hours for Aria + Trilinos Release and Dev
- Extensive testing and porting before the branch of Trilinos 9.0
- Upgrade to Trilinos 9.0 went very smoothly
 - Transition to Trilinos 9.0 was done in less than one week (could have been done in one day)



SIERRA + Trilinos Integration: STANA Website

<http://sierra-trac.sandia.gov/trac/sierra/wiki/Modules/Aria/SubProjects/STANA>



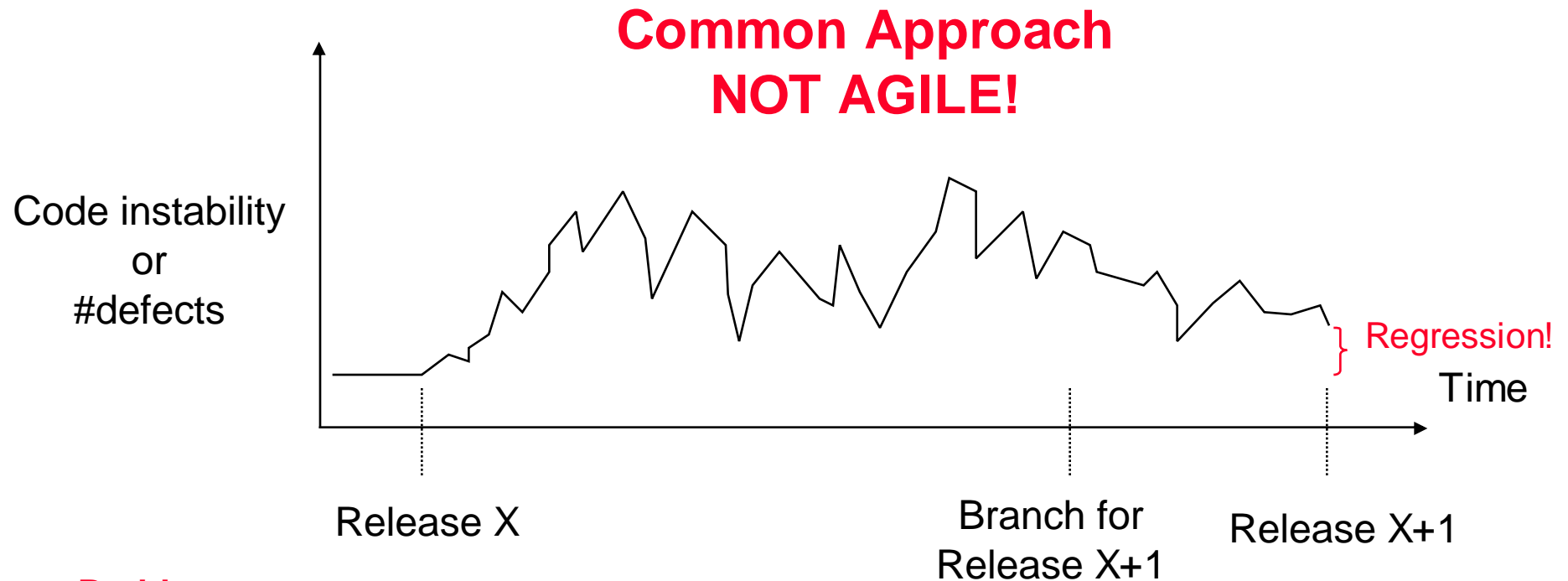
Lean/Agile Software Engineering Principles

- High quality software is developed in small increments and with sufficient testing in between sets of changes.
- High quality defect-free software is most effectively developed by not putting defects into the software in the first place (i.e. code reviews, pair programming, etc.).
- High quality software is developed in short fixed-time iterations.
- Software should be delivered to real (or as real as we can make them) customers in short intervals.
- Ruthlessly remove duplication in all areas.
- Avoid points of synchronization. Allow people to work as independently as possible and have the system set up to automatically support this.
- Most mistakes that people make are due to a faulty process/system (W. Edwards Deming).
- Automation is needed to avoid mistakes and improve software quality.

References: <http://www.cs.sandia.gov/~rabartl/readingList.html>



Lean/Agile Methods: Development Stability



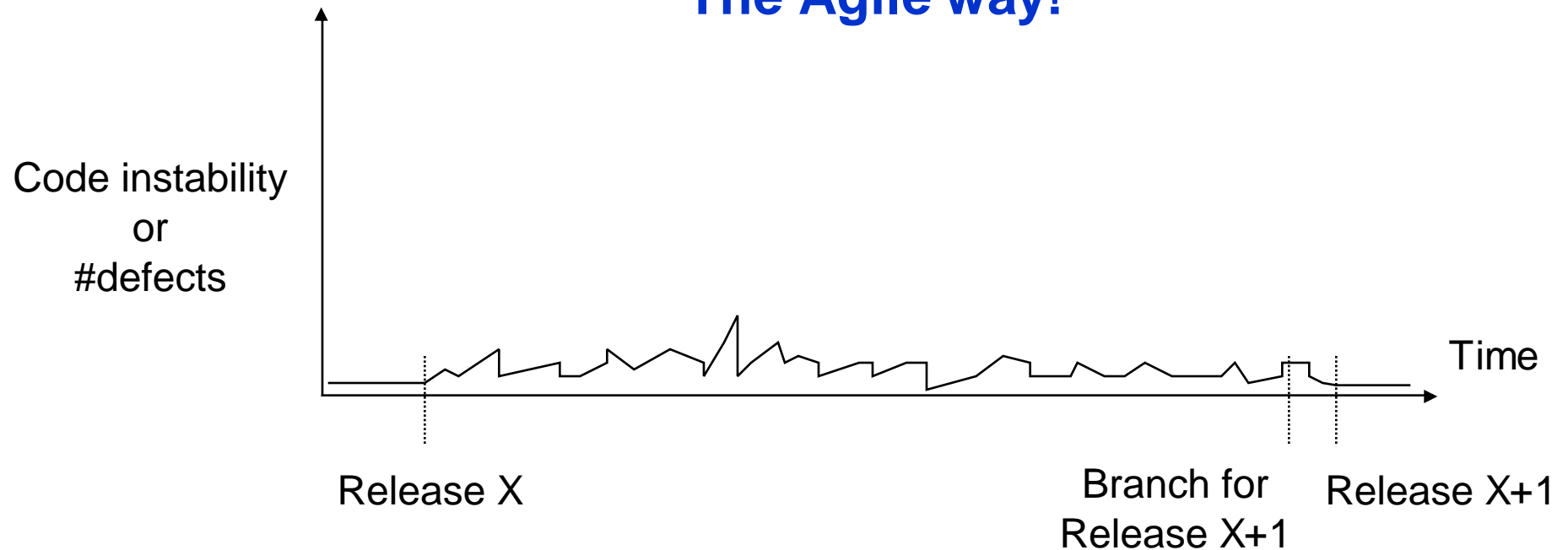
Problems

- Cost of fixing defects increases the longer they exist in the code
- Difficult to sustain development productivity
- Broken code begets broken code (i.e. broken window phenomenon)
- Long time between branch and release
 - Difficult to merge changes back into main development branch
 - Temptation to add “features” to the release branch before a release
- High risk of creating a regression



Lean/Agile Methods: Development Stability

The Agile way!

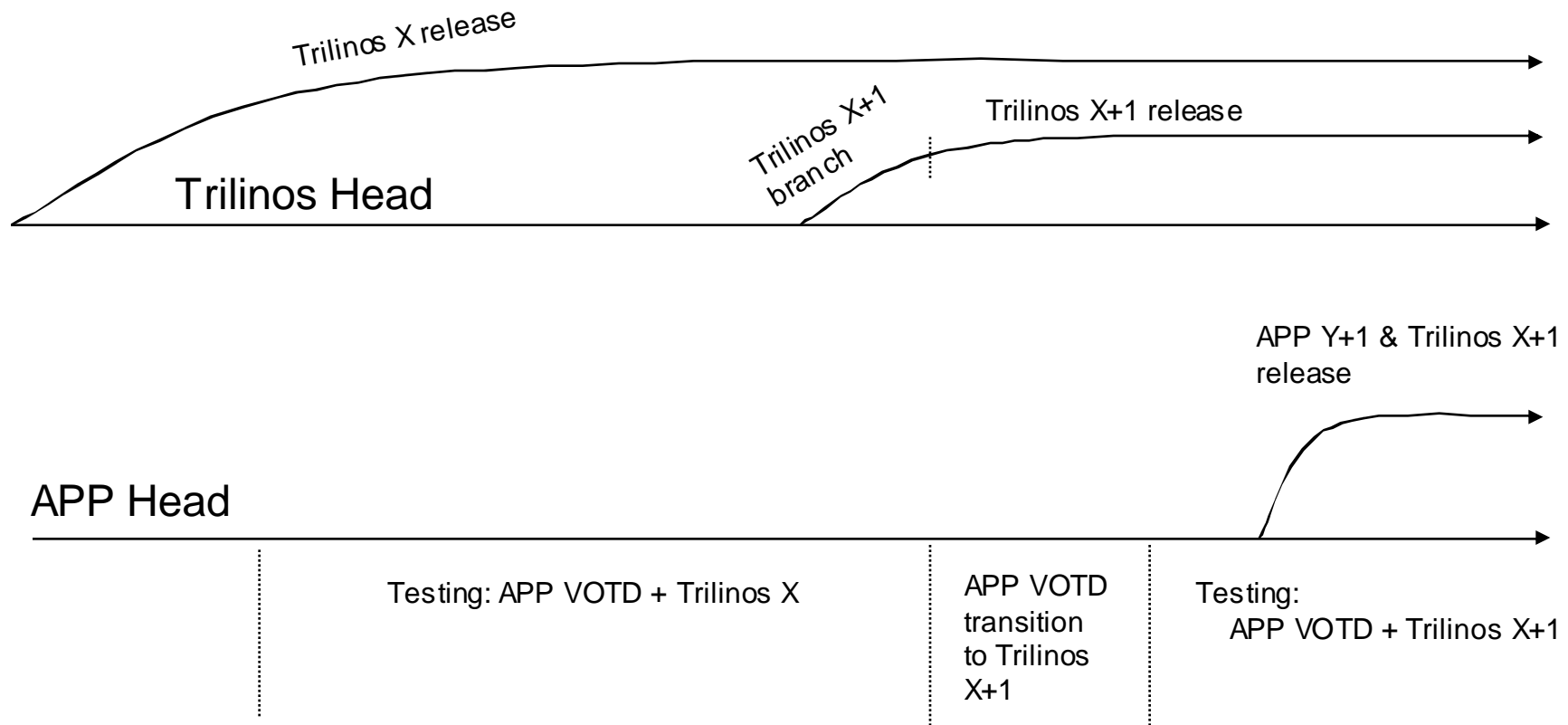


Advantages

- Defects are kept out of the code in the first place
- Code is kept in a near releasable state at all times
- Shorten time needed to put out a release
- Allow for more frequent releases
- Reduce risk of creating regressions
- Decrease overall development cost



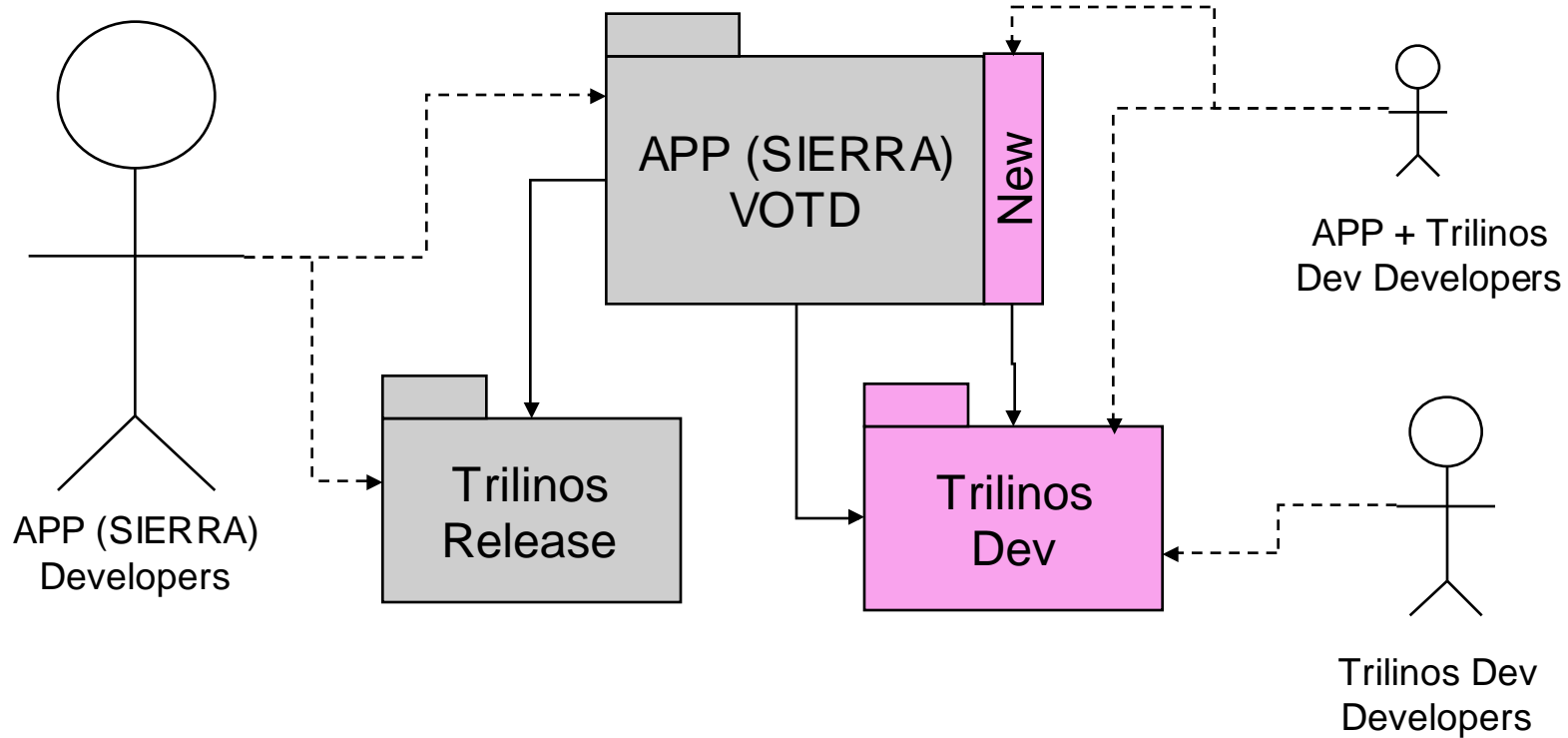
APP Only Upgrades After Each Major Release of Trilinos



- Transition from Trilinos X to Trilinos 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



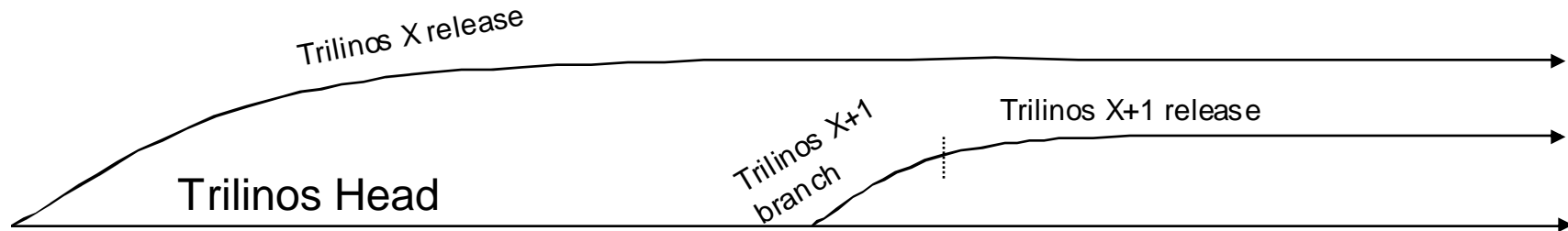
APP Builds Against both Trilinos Release and Trilinos Dev



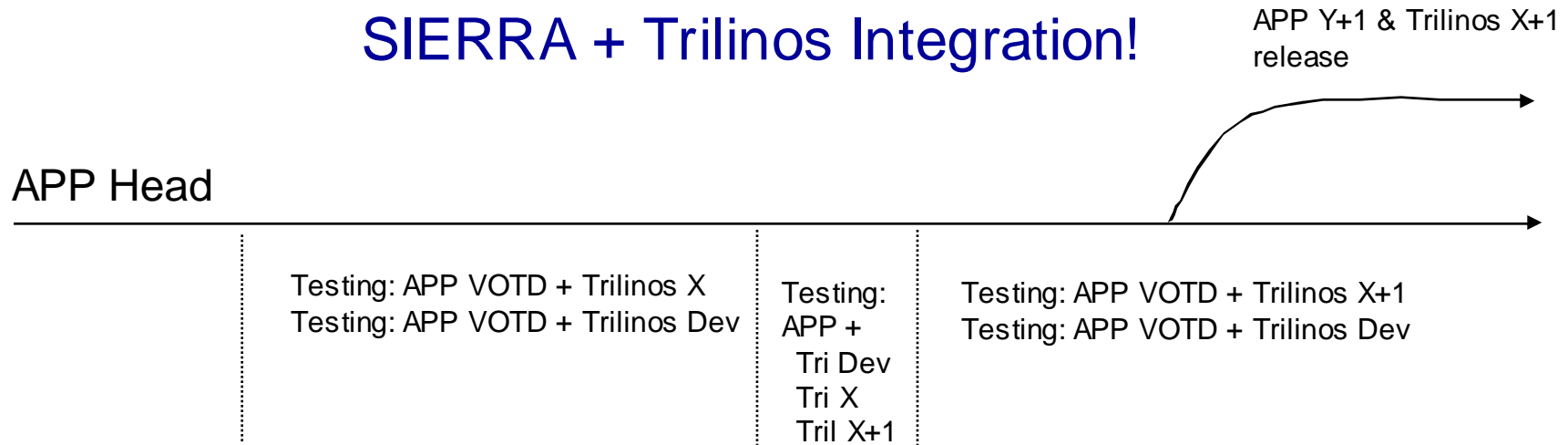
- APP (SIERRA) VOTD Developers only build/test against Trilinos Release
- Changes between Trilinos Release and Trilinos Dev handed through:
 - Refactoring
 - Minimal ifdefs (NO BRANCHES)!
- Trilinos Dev Developers work independent from APP
- Use of staggered releases of Trilinos and APP
- APP + Trilinos Dev Developers drive new capabilities



APP Builds Against both Trilinos Release and Trilinos Dev



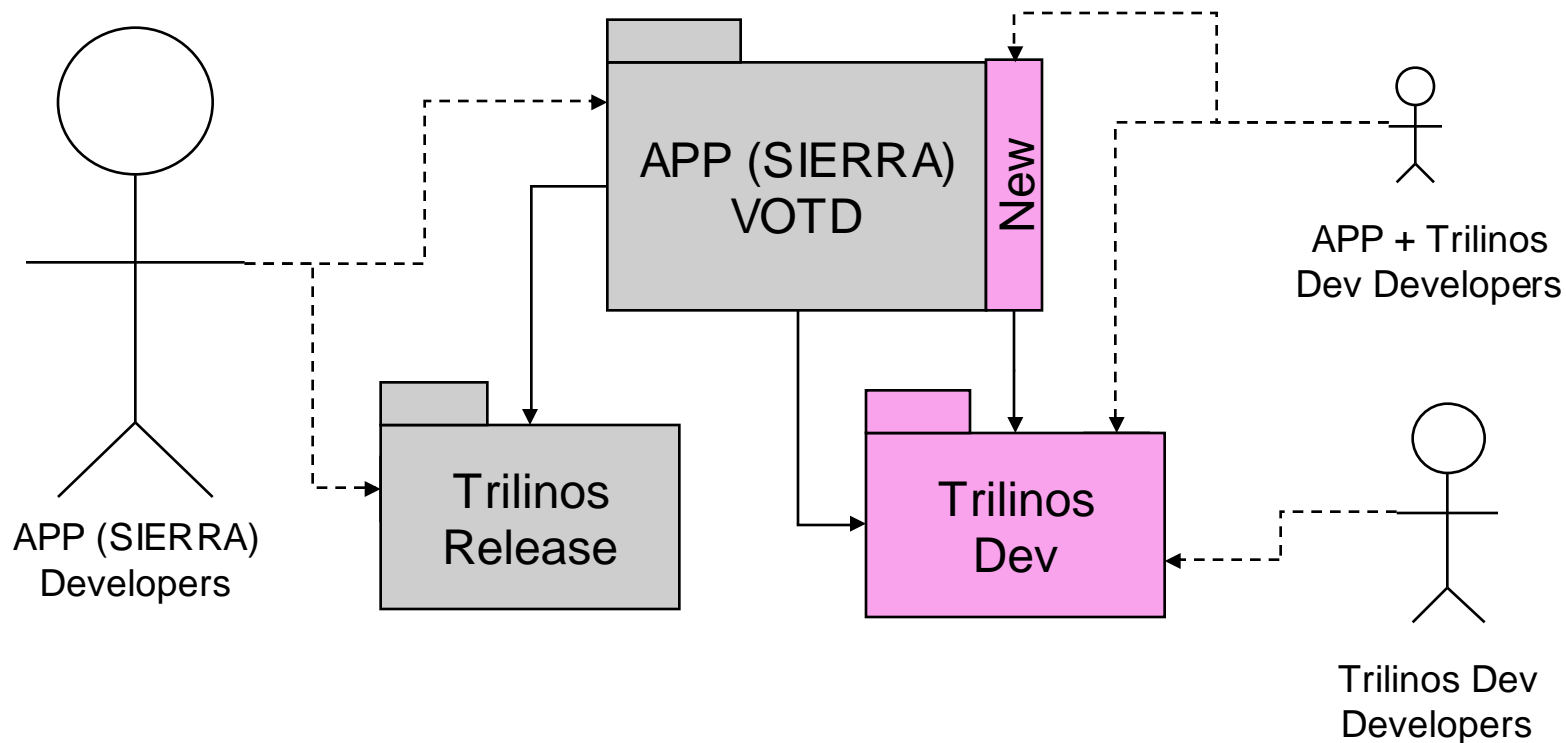
SIERRA + Trilinos Integration!



- All changes are tested in small batches
- Low probability of experiencing a regression
- Extra computing resources to test against 2 (3) versions of Trilinos
- Some difficulty flagging regressions of APP + Trilinos Dev
- APP developers often break APP + Trilinos Dev
- Difficult for APP to have rely on Trilinos too much
- Hard to verify Trilinos for APP before APP release



Challenges of APP + Trilinos Release and Dev Integration



Problems

- APP developers sometimes break APP + Trilinos Dev New
- APP + Trilinos Dev inherits instability of APP and Trilinos development lines

Improvements

- Make Trilinos Dev backward compatible with Trilinos Release
=> Minimize need to refactor and ifdef
- Improve stability of Trilinos Dev
- Improve stability of APP VOTD



SIERRA + Trilinos Integration: Opportunities and Challenges

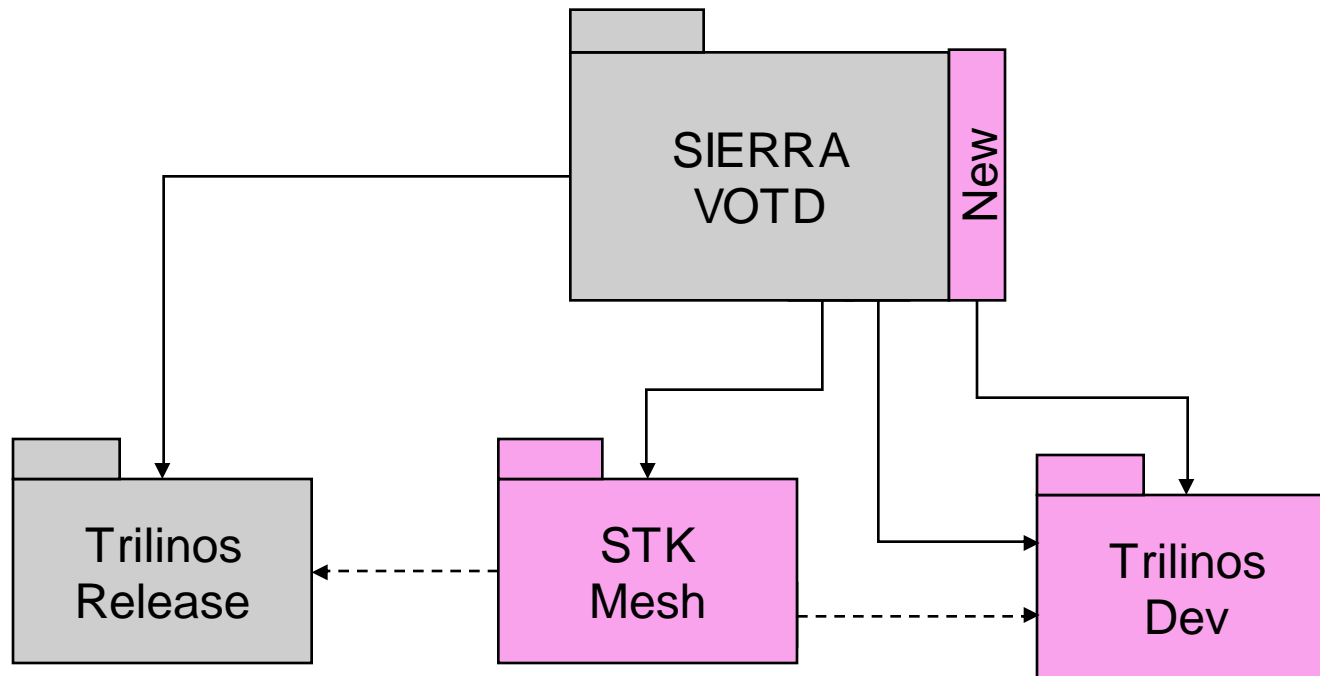
- SIERRA Framework Developers would like to consider tighter integration with Trilinos:
 - Move new SIERRA toolkits packages into Trilinos
 - STK_Mesh
 - STK_IO?

=> Make these available for rapid production and other projects
 - Develop the FEI through Trilinos instead of a SIERRA TPL
 - => Allow FEI to be updated more frequently
 - Replace SIERRA code with Trilinos code:
 - Teuchos::ParameterList
 - Intrepid
 - Phalanx

=> Reduce duplication and increase Trilinos impact
- Challenge: Tighter integration of APP and Trilinos does not fit well into current APP + Trilinos Release and Dev model!



APP + Trilinos Integration: Problems with Tighter Integration



Approach?

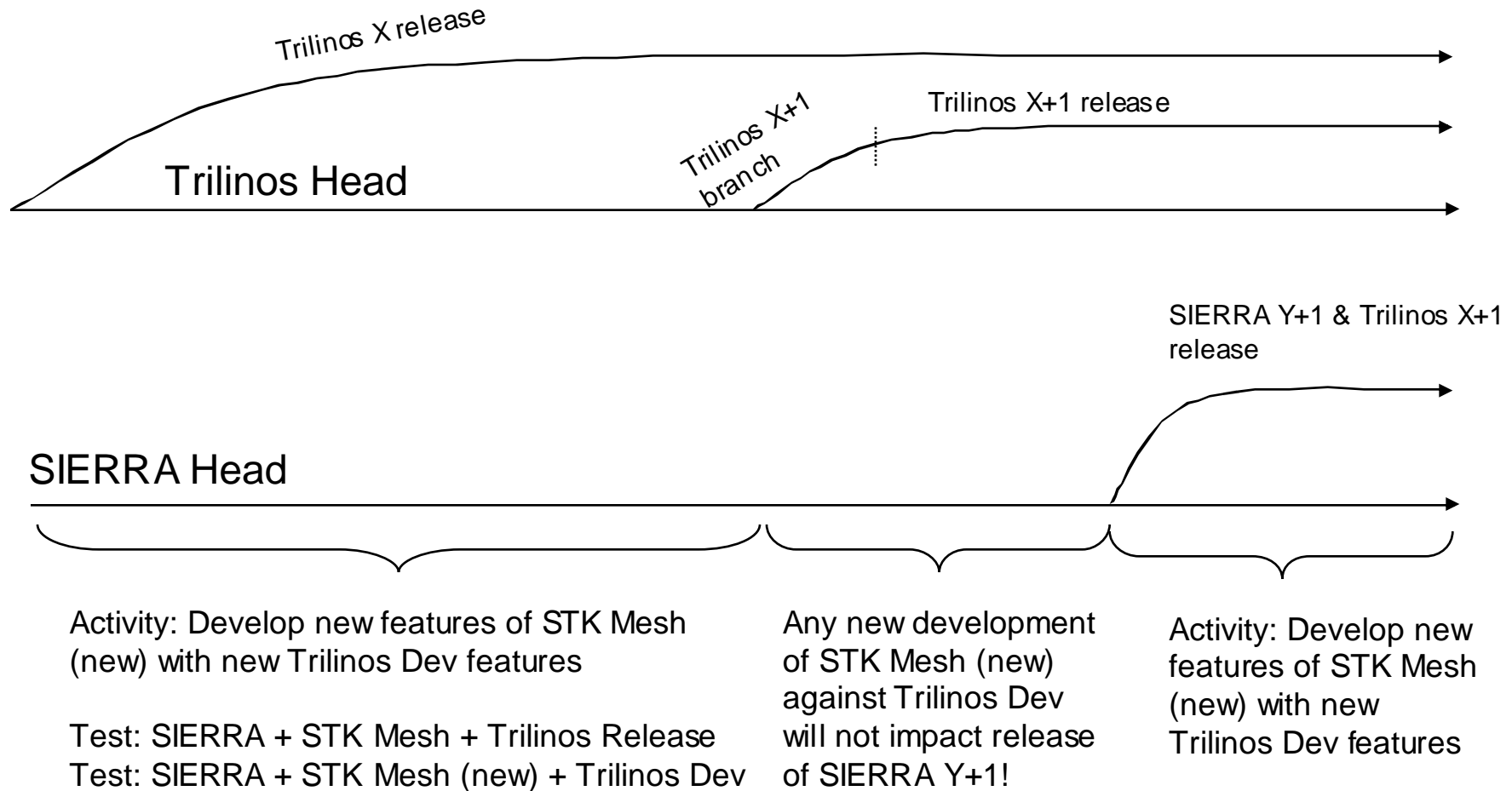
- Check out STK Mesh from Trilinos separately to build with SIERRA?
- Ifdef STK Mesh to build against both Trilinos Release and Trilinos Dev?

Problems

- Development of STK Mesh requires new features in Trilinos packages (i.e. Teuchos)
- STK Mesh built against Trilinos Release will not have some features!



APP + Trilinos Integration: Problems with Tighter Integration



Conclusion: This will be complex and involve greater risk!

Is there another way?

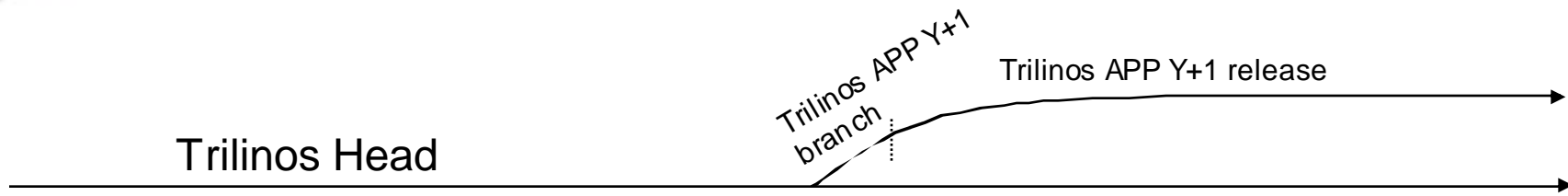


APP + Trilinos Integration: Different Collaboration Models

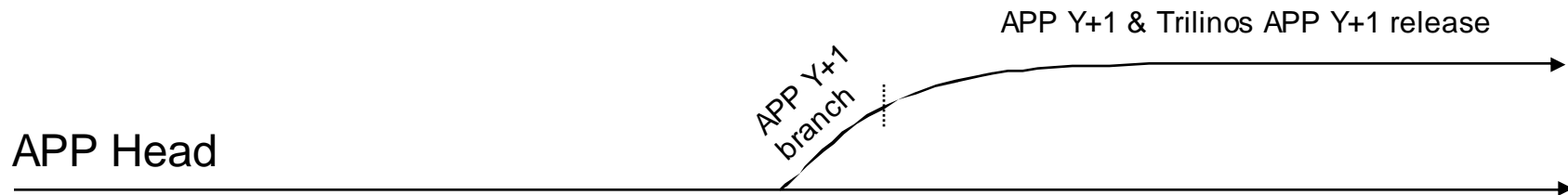
- APP only upgrades after each major release of Trilinos
 - Little to no testing of APP + Trilinos Dev in between versions
- APP builds against both Trilinos Release and Trilinos Dev
 - APP developers work against Trilinos Release
 - APP + Trilinos team(s) build against Trilinos Dev
 - Nightly and continuous integration testing done for both APP + Trilinos Release and Dev
 - Must handled staggered releases of Trilinos and APP
- APP developed only against Trilinos Dev
 - APP developers work directly against Trilinos Dev checked out every day
 - Releases best handled as combined releases of APP and Trilinos



APP developed only against Trilinos Dev



Future of SIERRA + Trilinos Integration?



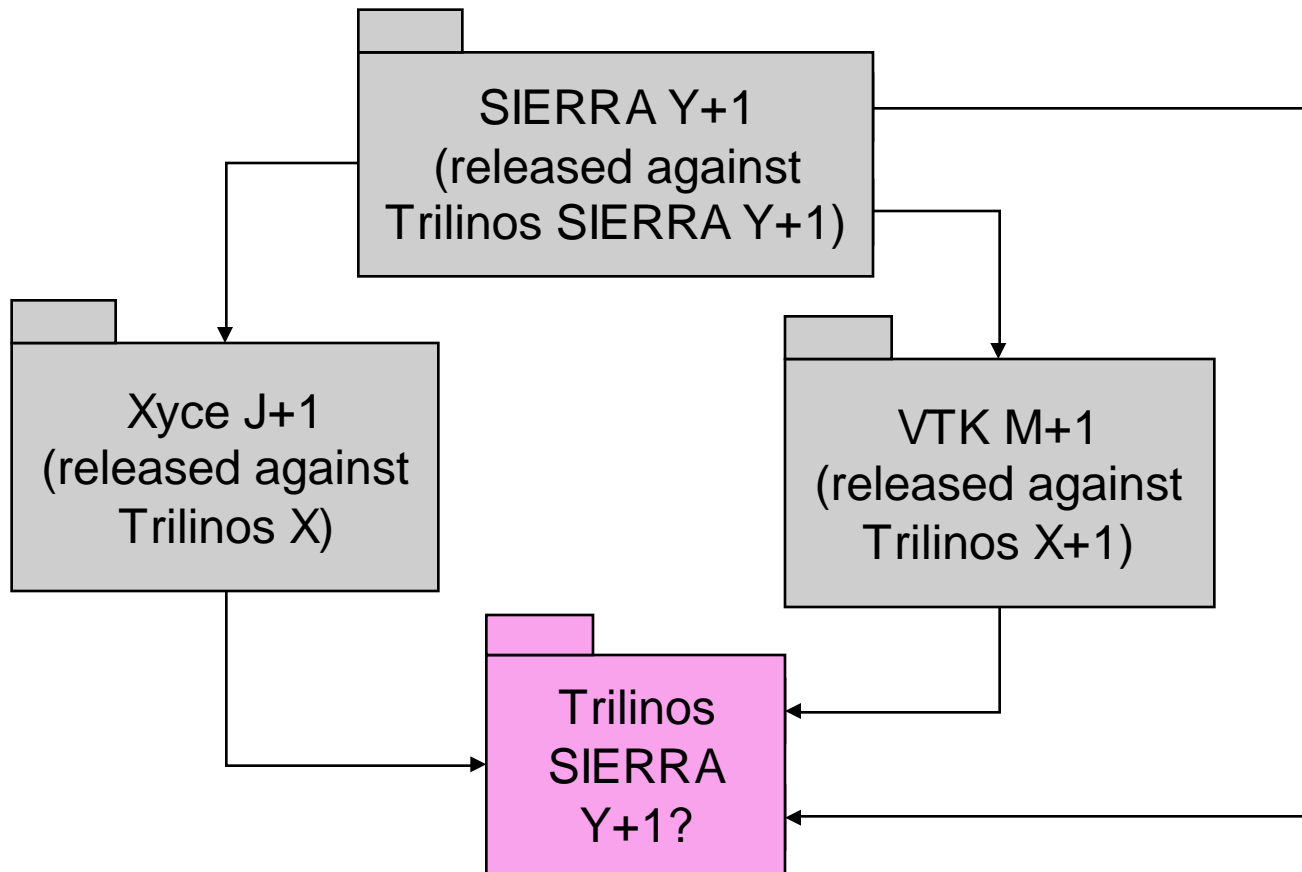
Testing: APP VOTD + Trilinos Dev

Supported with continuous integration testing!

- All changes are tested in small batches
- Low probability of experiencing a regression
- Less computing resources for testing
- Regressions and flagged immediately by APP developers
- Can support tighter integration efforts
- Supports rapid development of new capability from top to bottom
- Requires Trilinos to be more stable
- Other issues arise as well



Challenges with APP-Specific Trilinos Releases



Multiple releases of Trilinos presents a possible problem with complex applications

Solution:

=> Provide perfect backward compatibility of Trilinos X through Trilinos SIERRA Y+1



APP + Trilinos Continuous Integration: Solutions

- **Proposed approach:**
 - Develop APP VOTD directly against Trilinos Dev (not against Trilinos Release)
 - Create special releases of Trilinos just for these APPs
 - APP-specific releases of Trilinos will only be needed for these special APPs where tighter integration is required
 - Protect development work with continuous integration server and feedback
- **Improvements to Trilinos needed to support this:**
 - Improve the stability of “Stable” code in Trilinos Dev (see later presentation)
 - Preserve perfect backward compatibility for Trilinos for some period of time
 - => Allows some flexibility of what version of Trilinos gets used by customer codes
 - Improve other related software engineering practices

See the talk:

“Maintaining stability of Trilinos Dev - Stable vs Experimental Code”