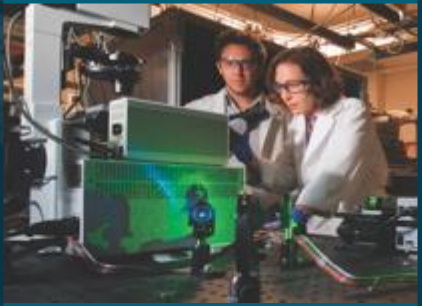
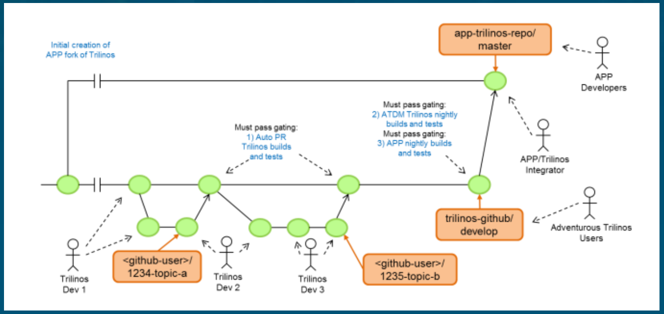
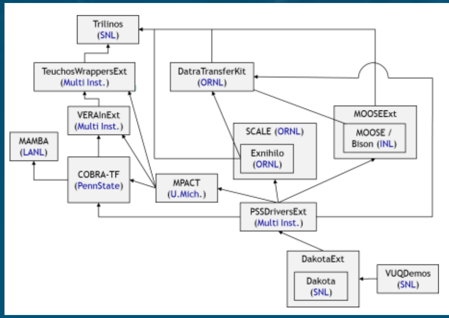
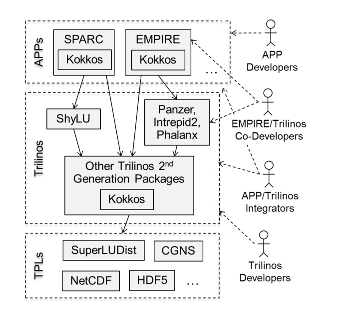
Development and Integration Workflows for Large Complex Distributed CSE Software Efforts









PRESENTED BY

Roscoe A. Bartlett, Ph.D.
Dept. Software Engineering & Research
https://bartlettroscoe.github.io

SAND2019-2092 C

Overview of CASL

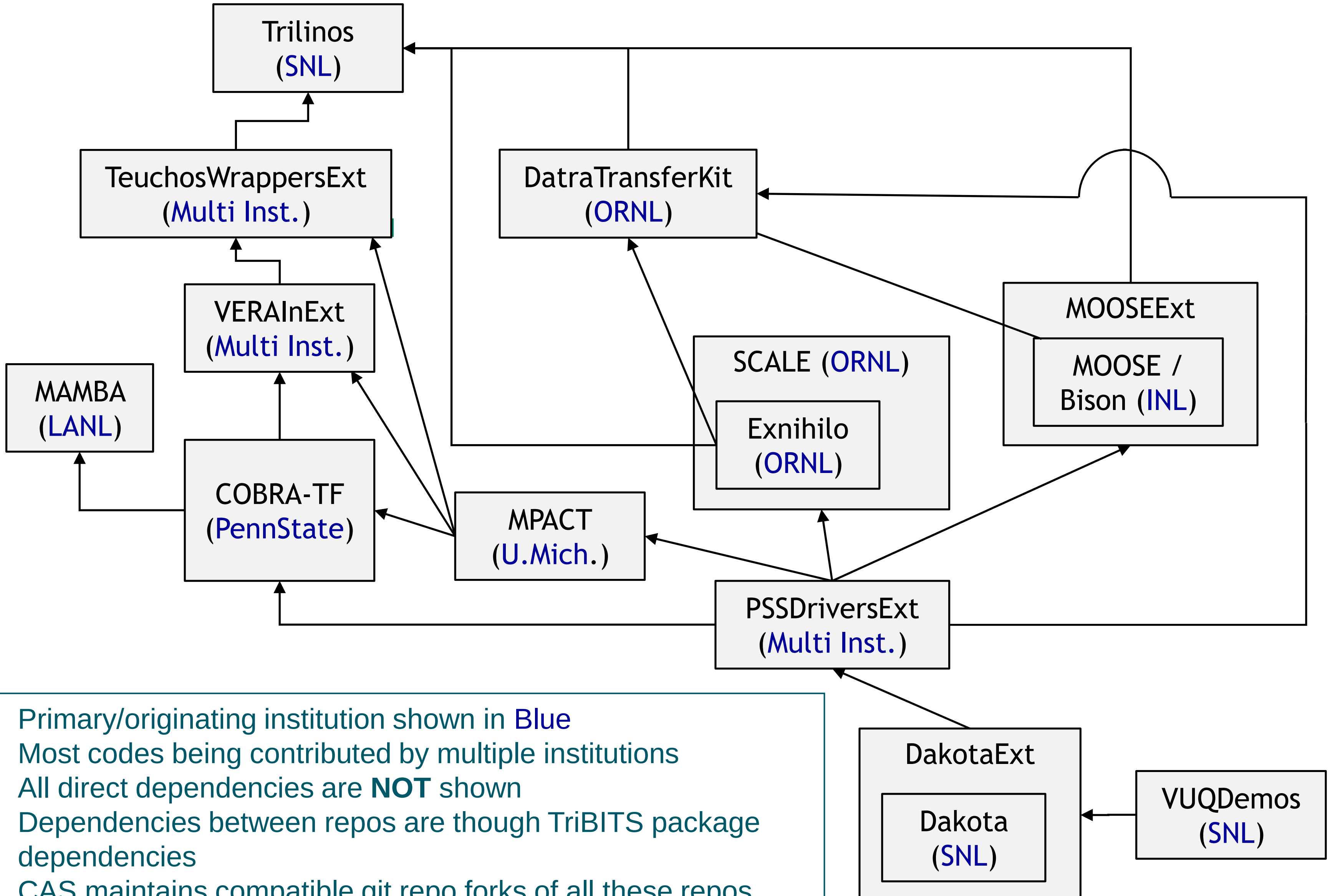
- **CASL: C**onsortium for the **A**dvanced **S**imulation of **L**ightwater reactors
- DOE Innovation Hub including DOE labs, universities, and industry partners
- Goals:
  - Advance modeling and simulation of lightwater nuclear reactors
  - Produce a set of simulation tools to model lightwater nuclear reactor cores to provide to the nuclear industry: **VERA: Virtual Environment for Reactor Applications**.
- Phase 1: July 2010 – June 2015
- Phase 2: July 2015 – June 2020
- Organization and management:
  - ORNL is the hub of the Hub
  - Milestone driven (6 month plan-of-records (PoRs))
  - Focus areas: **Physics Integration (PHI)**, Thermal Hydraulic Methods (THM), Radiation Transport Methods (RTM), Advanced Modeling Applications (AMA), Materials Performance and Optimization (MPO), Validation and Uncertainty Quantification (VUQ)

Roscoe Bartlett was PHI software engineering and integration lead for CASL from 2010-2016.

CASL VERA Development & Integration Overview (≈2016)

- VERA development is complicated!
- VERA Currently Composed of:
  - 21 different repositories on casl-dev.ornl.gov (some git clones of other repos) most with a different access list (NDAs, Export Control, IP, etc.)
  - Integrating development efforts from many teams from 9+ institutions
- Large single CMake build system using TriBITS CMake Framework:
  - Very large full source code base:
    - 55K source and script files
    - 12M lines of code (not comments)
    - 2,700 CMakeLists.txt files
  - 229 packages + subpackages enabled (out of 496 total) ≈ **46% of full code base**
  - Most CMake developer reconfigures take place in less than 30 seconds!
- VERA Software Development Process:
  - VERA integration maintained by continuous and nightly testing:
    - Pre-push CI testing: checkin-test-vera.sh, cloned VERA git repos
    - Post-push CI testing: CTest/CDash, all VERA git repos
    - Nightly testing: MPI and Serial builds, Debug and Release builds, …
    - Maintain 100% passing builds and tests most days!
  - Many internal and external repository integrations on daily basis
  - VERA releases are taken off of stable ‘master’ branches on casl-dev git repos.
  - Low maintenance cost of the infrastructure

(Selected) CASL VERA Git Repositories (≈2015)



- Primary/originating institution shown in Blue
- Most codes being contributed by multiple institutions
- All direct dependencies are **NOT** shown
- Dependencies between repos are though TriBITS package dependencies
- CAS maintains compatible git repo forks of all these repos

Overview of SNL ATDM (2019)

Advanced Technology Development & Mitigation (ATDM) Project

- Started in FY14 under DOE Advanced Simulation and Computing (ASC) Program
- Consumed into the larger DOE Exascale Computing Project (ECP) in FY16
- Background/Motivation:
  - **Exascale computers coming in 2023** using **new programming models** and hardware that current generation of ASC CSE codes will not run.
  - Rapidly developing/changing pre-exascale hardware and system software
- Mission of ATDM:
  - Design **next-generation exascale CSE codes** unconstrainted by software.
  - Leverage components and advanced algorithms for sensitivity analysis, design optimization, calibration, inversion, UQ/QMU, etc.

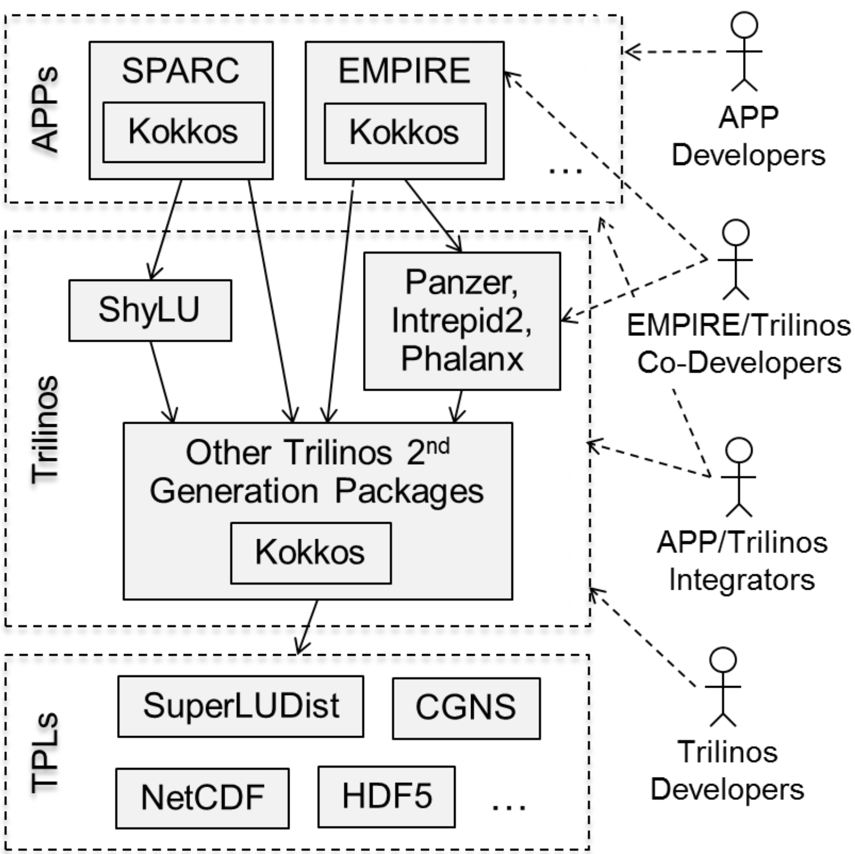
Sandia National Labs (SNL) ATDM Project:

- Two Primary SNL ATDM Codes:
  - **EMPIRE** – ElectroMagnetic Plasma In Radiation Environments
  - **SPARC -** Sandia Parallel Aerodynamics and Reentry Code
- Leveraging & co-developing **2nd Trilinos packages built on Kokkos** abstraction layer for on-node and on-GPU performance, Tpetra for node/GPU aware distributed memory linear algebra data-structures, and solvers built on these.

SNL ATDM Development and Integration Overview (2019)

Development & Integration Overview

- Core functionality provided by SNL 2nd generation Trilinos (Kokkos, Tpetra, etc.)
- SNL ATDM APP requirements drive Trilinos development.
- Each SNL ATDM APP maintains its own fork of Trilinos that is updated periodically.



Challenges

- Very long and expensive builds for templated Kokkos-based C++11 code.
- Limited computer testing resources.
- APPs needing frequent updates of Trilinos without getting new defects
- Keeping Trilinos & APPs working on changing ATDM/ECP platforms and environments.
- Defects in system software (e.g. compilers, MPI) slipping through system testing and instead being detected in Trilinos and APPs.
- Pushing for higher production-quality software from Ph.D. researchers.

Challenges in the Early Years of SNL ATDM (pre 2018)

- Trilinos Stability Problems:
  - No testing requirement before Trilinos developers pushed changes to the main 'develop' branch.
  - Many nightly builds submitting to CDash dashboard had many failing builds and failing tests that persisted for long periods of time.
    => Made it hard to see new defects
  - Little-to-no automated testing of Trilinos suite on ATDM pre-exascale platforms.

- ATDM APP developers and other staff members directly pulled from the main Trilinos 'develop' branch:
  - APP developers and other staff members often experienced broken builds.
  - Some important builds (e.g. CUDA on GPUs) often broken for significant lengths of time.

- Impact:
  - ➢ Lower ATDM APP developer productivity.
  - ➢ Lower confidence in Trilinos.
  - ➢ Avoidance depending on more Trilinos packages that absolutely required.

Development & Integration Challenges: CASL & ATDM

Common challenges in CASL VERA and SNL ATDM:

- Balancing speed of integration vs. stability of updates
- Coordination of different development teams
- Keeping build and testing infrastructure working both in external repos/projects and internal to the project

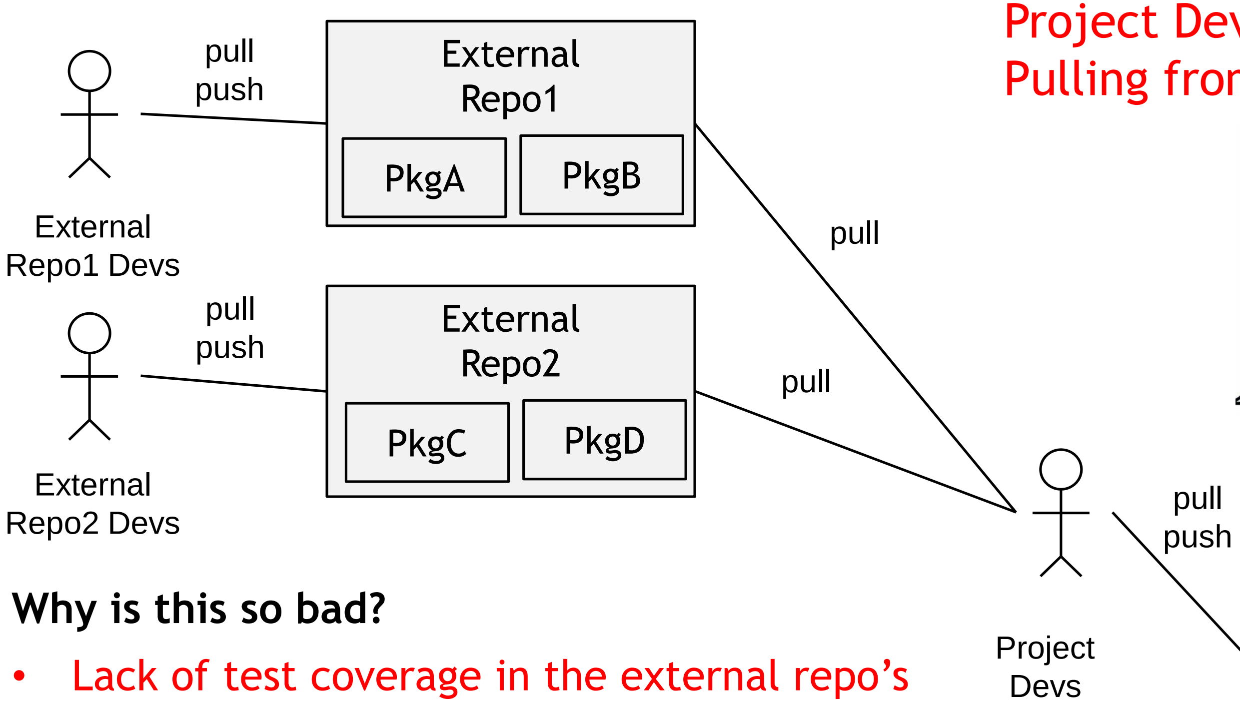
Different primary challenges in CASL VERA vs. SNL ATDM:

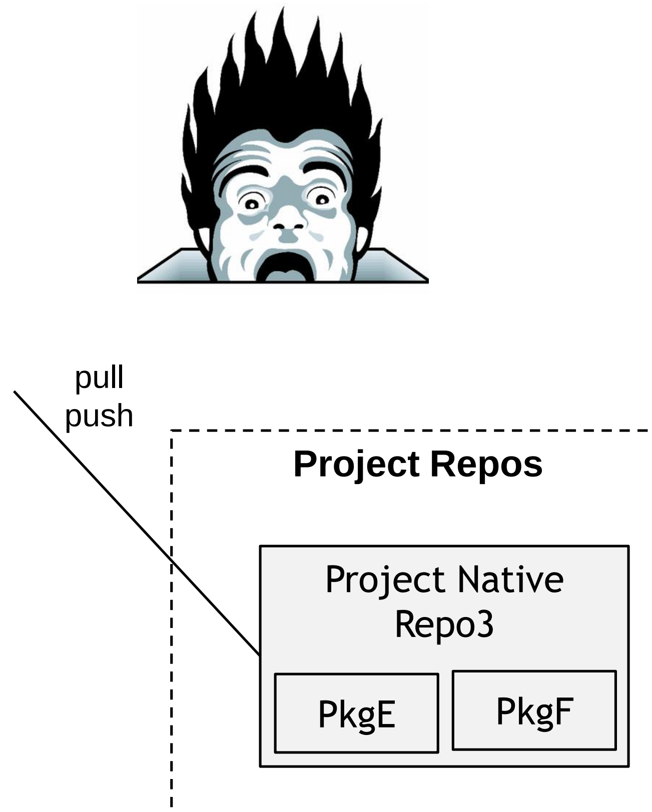
- **CASL VERA**:
  - Coordination of different development teams for multiple institutions.
  - Maintaining integrated build, test, and deployment from many different external projects.
- **SNL ATDM**:
  - Productive development and integration on many unstable buggy changing pre-exascale platforms.
  - Maintaining portability on wide range of ATDM/ECP platforms
  - Fast integration.

Multi-Team Multi-Repository Testing & Integration Basics

What Not to Do

Project Developers Directly Pulling from the External Repos

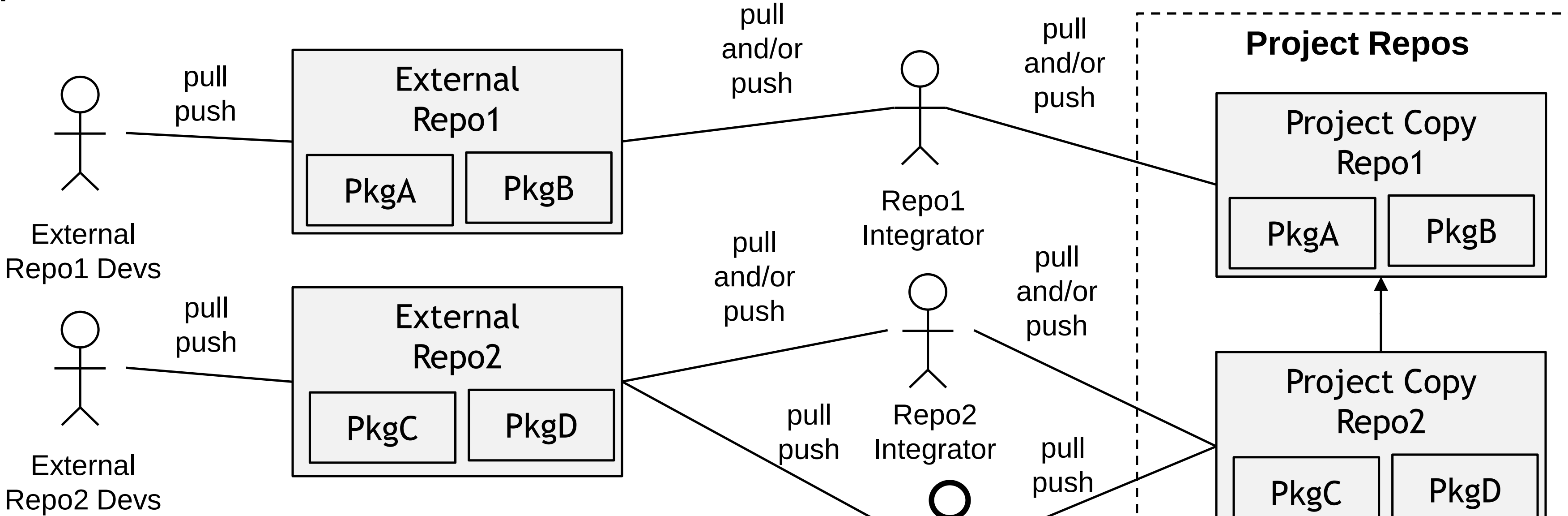


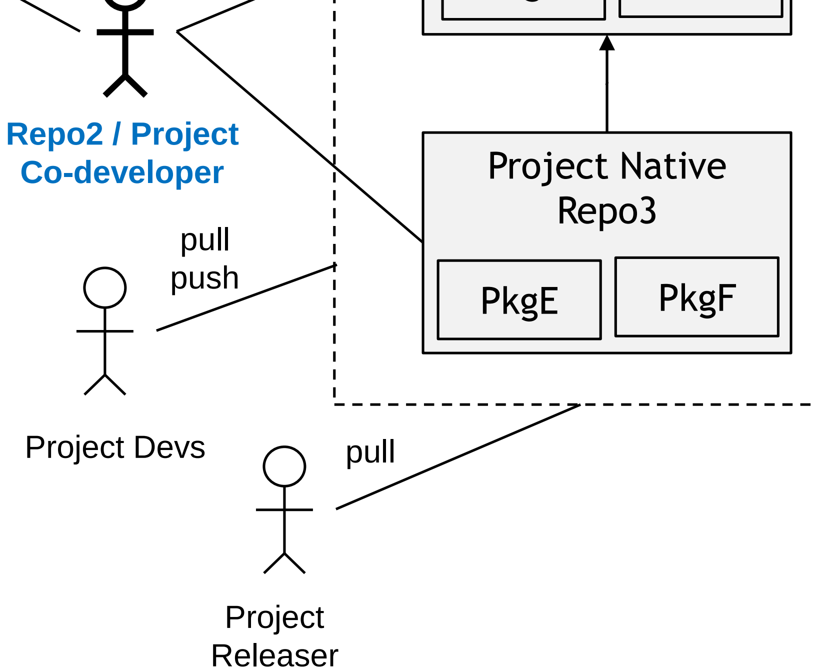


Why is this so bad?

- Lack of test coverage in the external repo's native test suite to cover project's needs.
- External repo developers not testing against the project's code and tests.
- External repo may be broken w.r.t. to the project for long period of time.
- Project developers frequently pull code that does not even configure or build.
- Broken code frequently interrupting the work of project developers.

Managing Internal and External Development & Integration





Project must keep consistent clones of every external repo and carefully sync updates!

Issues that need to be addressed:

- Flexibility for development inside and outside of the project.
- Providing a flow of frequent stable updates of the software.
- Maintaining the stability of the software to keep project developers productive.
- Making non-backward compatible changes across many repos.
- Full tracking of changes and updates.

Basic Parts to Development & Integration Process

- **Git Workflows:**
  - How git repositories and branches are set up, how merges occur, what git commands are run, etc.
  - Different git workflows used for external repo developers, Project developers, and repo/project co-developers.

- **Testing gates for workflows:**
  - Gating test suites can/should be run before each “merge” in the workflow.
  - Gating tests can be run manually or automated, daily or “every-so-often”.
  - Important test suites:

    RepoX build & tests: Gates updating the main RepoX development branch.

    Project builds & tests: Gates all updates of the project’s repos.

- **Detection, triage and fixing of new failing builds and tests:**
  - Detection and notification of new failures.
  - Triage failures.
  - Address failures.
  - Manage & follow-up.

Single External Repo Project Integration

Trilinos => ATDM APP

2) ATDM Trilinos Nightly Builds & Tests (CDash)

ATDM — 30 builds

Site	Build Name	Update		Configure			Build			Test						
		Revision	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail	Pass	Time ^	Proc Time	Start Time	Labels
sems-rhel6	Trilinos-atdm-sems-rhel6-gnu-opt-serial	254bd0	12s	0	19	1m 21s	0	29	30m 2s	0	0	1781	5m 13s	41m 31s	Oct 22, 2018 - 06:18 UTC	(25 labels)
sems-rhel6	Trilinos-atdm-sems-rhel6-gnu-opt-openmp	254bd0	12s	0	19	1m 25s	0	29	35m 53s	0	0	1891	5m 42s	46m 58s	Oct 22, 2018 - 05:29 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-intel-opt-serial	254bd0	18s	0	19	3m 44s	0	50	1h 26m 20s	0	0	1780	5m 48s	1h 22m 10s	Oct 22, 2018 - 10:03 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-intel-debug-serial	254bd0	12s	0	19	3m 30s	0	50	1h 26m 17s	0	0	1777	5m 59s	1h 23m 51s	Oct 22, 2018 - 08:25 UTC	(25 labels)
sems-rhel6	Trilinos-atdm-sems-rhel6-intel-opt-openmp	254bd0	12s	0	19	2m 10s	0	50	1h 39m 6s	0	0	1891	6m 1s	50m 13s	Oct 22, 2018 - 04:21 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-intel-opt-openmp	254bd0	18s	0	19	3m 43s	0	50	1h 29m 6s	0	0	1890	6m 24s	1h 31m 37s	Oct 22, 2018 - 11:04 UTC	(25 labels)
white	Trilinos-atdm-white-ride-gnu-opt-openmp	254bd0	12s	0	19	3m 34s	0	15	24m 50s	0	0	1890	6m 32s	1h 31m 27s	Oct 22, 2018 - 06:10 UTC	(25 labels)
waterman	Trilinos-atdm-waterman-gnu-opt-openmp	254bd0	24s	0	19	3m 20s	0	15	16m 58s	0	0	1890	8m 8s	2h 17m 32s	Oct 22, 2018 - 04:24 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-gnu-opt-openmp	254bd0	12s	0	19	2m 53s	0	4	32m 56s	0	0	1890	8m 38s	1h 41m 4s	Oct 22, 2018 - 12:45 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-intel-debug-openmp	254bd0	12s	0	19	3m 44s	0	50	1h 33m 30s	0	0	1889	12m 18s	3h 8m 15s	Oct 22, 2018 - 10:48 UTC	(25 labels)
serrano	Trilinos-atdm-serrano-intel-opt-openmp	254bd0	1m 18s	0	19	2m 27s	0	50	3h 24m 43s	0	0	1889	14m 20s	1h 47m 27s	Oct 22, 2018 - 04:25 UTC	(25 labels)
chama	Trilinos-atdm-chama-intel-debug-openmp	254bd0	2m 30s	0	19	5m 24s	0	50	7h 20m 11s	0	0	1890	14m 35s	1h 48m 42s	Oct 22, 2018 - 04:33 UTC	(25 labels)
chama	Trilinos-atdm-chama-intel-opt-openmp	254bd0	1m 48s	0	19	5m 19s	0	50	5h 38m 50s	0	0	1891	14m 53s	1h 51m 1s	Oct 22, 2018 - 04:36 UTC	(25 labels)
serrano	Trilinos-atdm-serrano-intel-debug-openmp	254bd0	1m 12s	0	19	2m 31s	0	50	4h 12m 54s	0	0	1888	15m 3s	1h 52m 43s	Oct 22, 2018 - 04:30 UTC	(25 labels)
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-KNL-panzer	254bd0	1m	0	1	4m 14s	0	50	1h 31m 21s	0	0	159	16m 18s	2h 8m 23s	Oct 22, 2018 - 14:29 UTC	Panzer
white	Trilinos-atdm-white-ride-gnu-debug-openmp	254bd0	18s	0	19	2m 34s	0	14	19m 16s	0	0	1886	16m 56s	4h 11m 29s	Oct 22, 2018 - 08:30 UTC	(25 labels)
sems-rhel6	Trilinos-atdm-sems-rhel6-gnu-debug-openmp	254bd0	18s	0	19	1m 26s	0	29	30m 33s	0	0	1890	17m 3s	2h 9m 50s	Oct 22, 2018 - 06:14 UTC	(25 labels)
sems-rhel6	Trilinos-atdm-sems-rhel6-gnu-debug-serial	254bd0	6s	0	19	1m 19s	0	29	29m 3s	0	0	1780	19m 15s	2h 14m 10s	Oct 22, 2018 - 04:30 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-gnu-debug-openmp	254bd0	12s	0	19	3m 10s	0	4	32m 39s	0	0	1890	19m 32s	4h 17m 36s	Oct 22, 2018 - 09:50 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-gnu-opt-serial	254bd0	12s	0	19	2m 34s	0	4	34m 14s	0	0	1776	21m 19s	5h 15m 19s	Oct 22, 2018 - 07:24 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-gnu-debug-serial	254bd0	12s	0	19	2m 57s	0	4	35m 26s	0	0	1776	30m 25s	6h 49m 21s	Oct 22, 2018 - 06:13 UTC	(25 labels)
white	Trilinos-atdm-white-ride-cuda-9.2-opt	254bd0	12s	0	20	4m 3s	0	50	1h 3m 10s	0	1	1896	33m 4s	4h 18m 17s	Oct 22, 2018 - 06:47 UTC	(25 labels)
white	Trilinos-atdm-white-ride-cuda-9.2-debug	254bd0	12s	0	20	3m 51s	0	50	1h 30m 31s	0	1	1895	46m 37s	6h 7m 22s	Oct 22, 2018 - 06:24 UTC	(25 labels)
waterman	Trilinos-atdm-waterman-cuda-9.2-opt	254bd0	12s	0	20	4m 4s	0	50	55m 3s	0	0	1900	49m 28s	6h 31m 26s	Oct 22, 2018 - 04:18 UTC	(25 labels)
waterman	Trilinos-atdm-waterman-cuda-9.2-debug	254bd0	24s	0	20	4m 25s	0	50	1h 25m 15s	0	0	1897	1h 18s	7h 55m 54s	Oct 22, 2018 - 04:56 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-debug	254bd0	12s	0	20	3m 48s	0	50	2h 44m 48s	0	1	1896	1h 13m 16s	9h 33m 11s	Oct 22, 2018 - 11:42 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-opt	254bd0	12s	0	20	3m 39s	0	50	1h 59m 21s	0	1	1897	1h 25m 11s	11h 8m 47s	Oct 22, 2018 - 12:40 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-opt	254bd0	18s	0	20	3m 34s	0	50	2h 1m 4s	0	1	1897	1h 27m 40s	11h 27m 54s	Oct 22, 2018 - 06:15 UTC	(25 labels)
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-debug	254bd0	18s	0	20	3m 52s	0	50	3h 10m 45s	0	1	1896	1h 32m 41s	12h 7m 18s	Oct 22, 2018 - 06:14 UTC	(25 labels)
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	254bd0	1m 6s	0	19	6m 33s	0	50	5h 52m 25s	0	3	1868	2h 39m 16s	21h 8m 4s	Oct 22, 2018 - 05:43 UTC	(25 labels)

Items per page All ▼

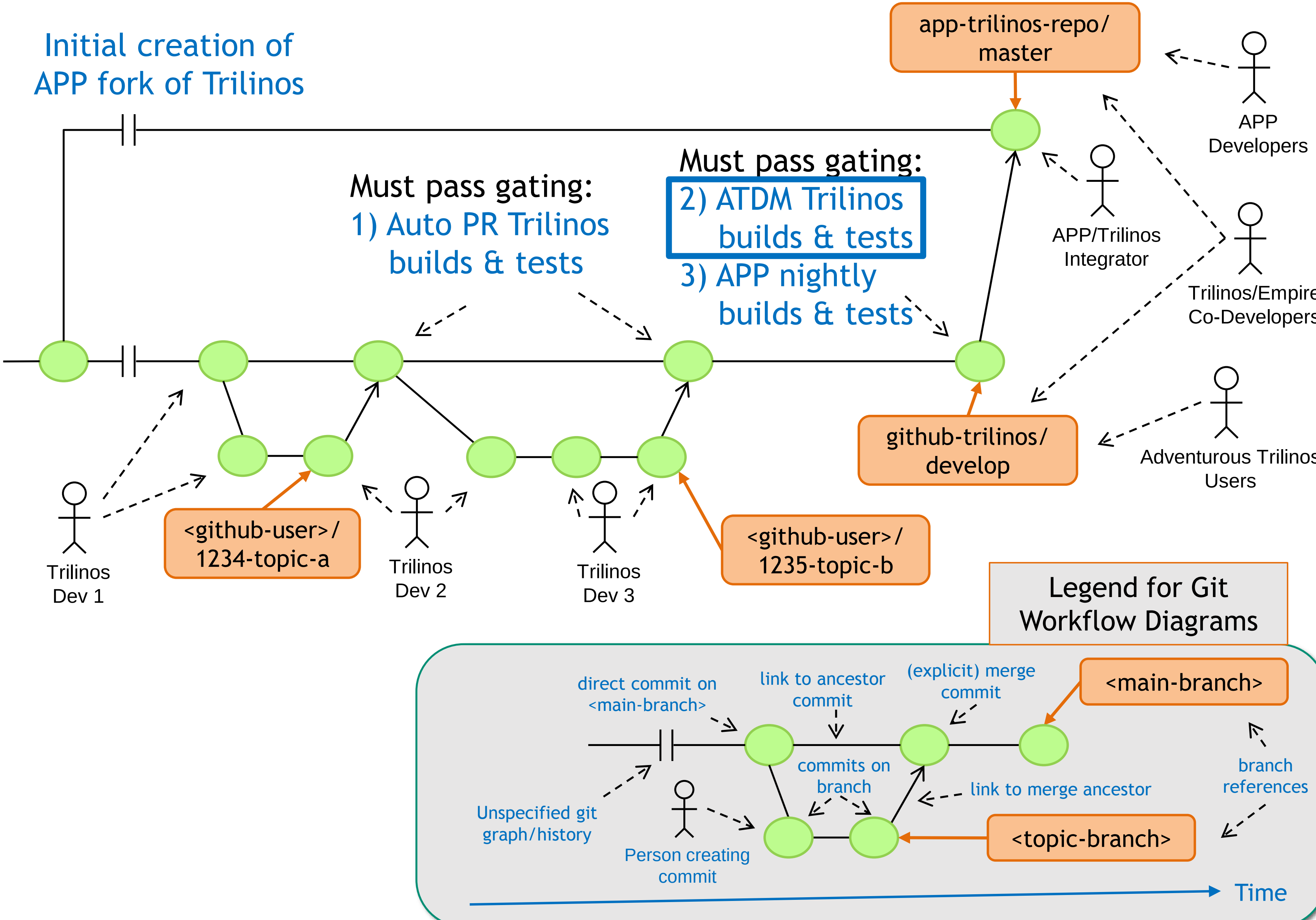
Specialized — 12 builds

Site	Build Name	Update		Configure			Build			Test						
		Revision	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail	Pass	Time ˅	Proc Time	Start Time	Labels
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-KNL	254bd0	1m	0	19	6m 14s	0	50	5h 54m 31s	0	10	1861	3h 53m 45s	1 day 6h 59m 52s	Oct 22, 2018 - 05:41 UTC	(25 labels)
ride	Trilinos-atdm-white-ride-cuda-9.2-debug-pt	254bd0	18s	0	10	5m 53s	2	50	2h 20m 23s	2	86	2717	1h 10m 17s	9h 19m 15s	Oct 22, 2018 - 04:23 UTC	(54 labels)
white	Trilinos-atdm-white-ride-cuda-9.2-debug-pt	254bd0	18s	0	10	5m 44s	2	50	2h 15m 54s	2	86	2717	1h 10m	9h 18m 19s	Oct 22, 2018 - 06:24 UTC	(54 labels)
waterman	Trilinos-atdm-waterman-cuda-9.2-release-debug	254bd0	24s	0	20	3m 50s	0	50	1h 22m 23s	0	1	1897	54m 30s	7h 11m 16s	Oct 22, 2018 - 06:09 UTC	(25 labels)
ride	Trilinos-atdm-white-ride-cuda-9.2-debug	254bd0	12s	0	20	3m 54s	0	50	1h 31m 20s	0	1	1895	46m 44s	6h 8m 19s	Oct 22, 2018 - 04:54 UTC	(25 labels)
ride	Trilinos-atdm-white-ride-cuda-9.2-opt	254bd0	12s	0	20	3m 36s	0	50	1h 3m 5s	0	1	1896	32m 43s	4h 16m 45s	Oct 22, 2018 - 04:22 UTC	(25 labels)
ride	Trilinos-atdm-white-ride-gnu-debug-openmp	254bd0	12s	0	19	3m 1s	0	14	19m 36s	0	0	1886	16m 57s	4h 12m 42s	Oct 22, 2018 - 04:24 UTC	(25 labels)
waterman	Trilinos-atdm-waterman-gnu-release-debug-openmp	254bd0	12s	0	19	3m 11s	0	21	21m 49s	0	1	1887	7m 35s	2h 8m 13s	Oct 22, 2018 - 07:29 UTC	(25 labels)
cee-rhel6	Trilinos-atdm-cee-rhel6-intel-opt-serial	254bd0	6s	0	0	2m 9s	0	50	1h 4m 48s	0	2	2004	6m 56s	54m 50s	Oct 22, 2018 - 07:38 UTC	(27 labels)
ride	Trilinos-atdm-white-ride-gnu-opt-openmp	254bd0	18s	0	19	2m 44s	0	15	24m 45s	0	0	1890	6m 27s	1h 31m 2s	Oct 22, 2018 - 04:17 UTC	(25 labels)
cee-rhel6	Trilinos-atdm-cee-rhel6-clang-opt-serial	254bd0	12s	0	0	1m 26s	0	50	36m 23s	0	4	2002	5m 50s	44m 14s	Oct 22, 2018 - 06:10 UTC	(27 labels)
cee-rhel6	Trilinos-atdm-cee-rhel6-gnu-opt-serial	254bd0	12s	0	0	1m 22s	0	4	35m 15s	0	2	2006	5m 48s	43m 52s	Oct 22, 2018 - 06:55 UTC	(27 labels)

Items per page All ▼

- Build and run native Trilinos test suite on all the ATDM platforms.
- First step in providing stable portability on many pre-exascale platforms.
- Builds are too expensive to run more than one set per 24-hour day.
- Frequent random system failures make detection of new code-related failures difficult.

ATDM Trilinos Development and Integration Workflows

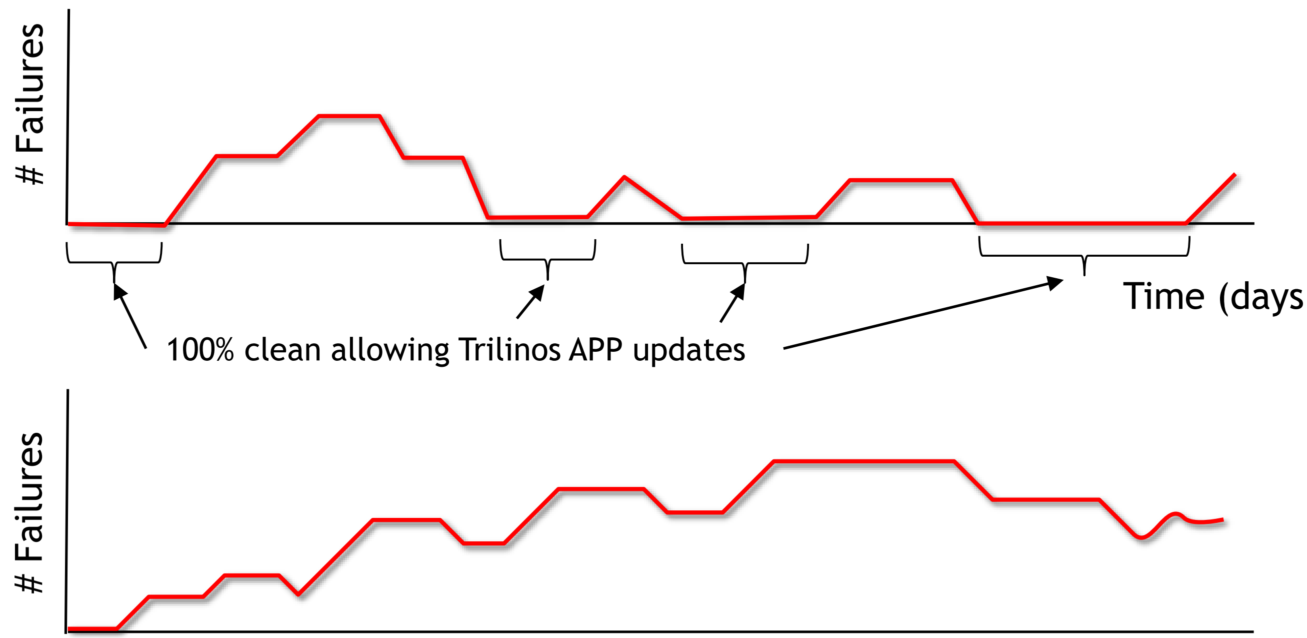


Injecting New Failures and Fixing Failures: A Race!

- **Mean-time to fail:** Average time (in days) for when a new failure shows up in ‘develop’ branch in one or more promoted ATDM Trilinos builds.
- **Mean-time to fix:** Average time (in days) to discover, triage and fix a failure on the Trilinos ‘develop’ branch in the promoted ATDM Trilinos builds.
- **The core problem:** If “mean-time to fail” is less than “mean-time to fix”, then the ATDM Trilinos builds on ‘develop’ on average will **ALWAYS be broken** (and therefore block updates of Trilinos to the APP customers)!

Mean-time to fix
<
Mean-time to fail

Mean-time to fix
>
Mean-time to fail

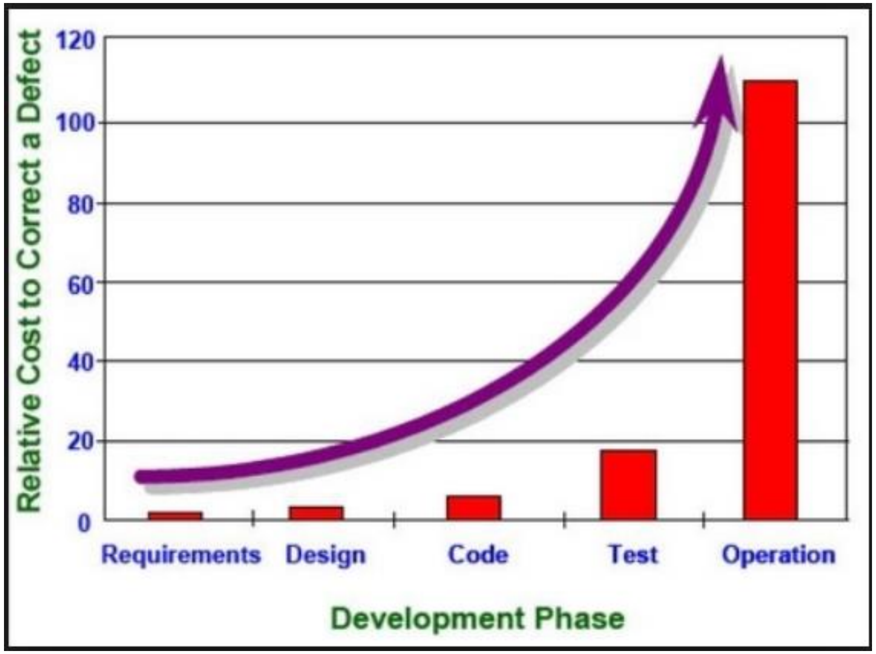


Reducing Time to Detect, Triage, and Address New Failures

ATDM Trilinos Builds & Tests

General SE Principles for Defects

- **Cost of a defect goes up (significantly) the longer it takes to detect and correct a defect.**



- **Lean/Agile SE Practices for dealing with defects:**
  - Strong automated testing (have tests help new detect defects)
  - Continuous testing (reduce the time to detect new defects caught by tests)
  - Continuous integration (reduce time to detect conflict defects)
  - **STOP THE LINE** when a new defect gets into the main development branch
    - Fixing defects in previously working software is higher priority than developing new features!

Detecting New Failures/Missing Results: CDash Email

FAILED (bm=1, twoif=2, twip=1, twif=2): Promoted ATDM Trilinos Builds on 2019-01-04

Builds on CDash (num/expected=33/33)
Non-passing Tests on CDash (num=4)

Builds Missing: bm=1
Tests without issue trackers Failed: twoif=2
Tests with issue trackers Passed: twip=1
Tests with issue trackers Failed: twif=2

Failures in **red** may require triage!

- **Missing test results!**
- **Failing tests without issue trackers!**

Builds Missing: bm=1

Group	Site	Build Name	Missing Status
ATDM	waterman	Trilinos-atdm-waterman-cuda-9.2-release-debug	Build exists but no test results

Tests without issue trackers Failed (limited to 20): twoif=2

Site	Build Name	Test Name	Status	Details	Consecutive Non-pass Days	Non-pass Last 30 Days	Pass Last 30 Days	Issue Tracker
sems-rhel6	Trilinos-atdm-sems-rhel6-intel-opt-openmp	Belos_BlockGmresPoly_Epetra_File_Ex_0_MPI_4	Failed	Completed (Failed)	1	1	22	

...

Tests with issue trackers Failed: twif=2

Site	Build Name	Test Name	Status	Details	Consecutive Non-pass Days	Non-pass Last 30 Days	Pass Last 30 Days	Issue Tracker
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	Anasazi_Epetra_BKS_norestart_test_MPI_4	Failed	Completed (Failed)	21	21	3	#3499

Reproducing ATDM Trilinos Builds: GitHub Issue

GitHub, Inc. [US] | https://github.com/trilinos/Trilinos/issues/3681

Steps to Reproduce

One should be able to reproduce this failure on waterman as described in:

- https://github.com/trilinos/Trilinos/blob/develop/cmake/std/atdm/README.md
  More specifically, the commands given for waterman are provided at:
- https://github.com/trilinos/Trilinos/blob/develop/cmake/std/atdm/README.md#waterman
  The exact commands to reproduce this issue should be:

```
$ cd <some_build_dir>/

$ source $TRILINOS_DIR/cmake/std/atdm/load-env.sh cuda-9.2-release-debug

$ cmake \
  -GNinja \
  -DTrilinos_CONFIGURE_OPTIONS_FILE:STRING=cmake/std/atdm/ATDMDevEnv.cmake \
  -DTrilinos_ENABLE_TESTS=ON -DTrilinos_ENABLE_Intrepid2=ON \
  $TRILINOS_DIR

$ make NP=20

$ bsub -x -Is -n 20 ctest -j20
```

How to Address Failures?

Already cleaned-up promoted builds clean:

a) Fix the failures => **Best option!**

b) Mark failing tests as "allow to fail" and not trigger global FAIL:

- Only for non-blocking issues
- Allows us to watch test run but not block updates of Trilinos to APPs
- Best for when someone is working to fix non-blocking failures.

c) (Temporarily) disable failing tests:

- Only for non-blocking issues
- Best for cases where no-one is going to work on fixing the failures soon.

d) Revert the commit(s) (or PR merge) causing the failure:

- Best for critical/blocking failures that can't be fixed ASAP.

Initial failures setting up new platforms:

a) Fix the failures

b) (Temporarily) disable failing tests

c) Mark failing tests as "allow to fail" and not trigger global FAIL

- NOTE: Reverting commits is NOT an option for cleaning up failures that occur when setting up new builds on new platforms or envs on existing platforms.

Conclusions and Lesions Learned CASL & SNL ATDM

- **Projects must set up their own forks of external repos that must be frequently updated and define integration testing workflows**
- **Detecting, Traiging, and Addressing New Failures:**
  - Running tests using similar configurations on different systems and compilers helps to speed up detection of new software defects.
  - Effective detection and triaging requires an analysis tool that takes a broad view of build and tests results to show trends, commonality, and history.
  - Likely 90-95% of failing (Trilinos) tests don't indicate a problem impacting a specific customer but they hide the 5-10% that do.
  - Must carefully scrutinize every failing test to detect new defects.
  - Must not allow existing failures to hide new failures!
- **Build and Test Systems:**
  - Heterogenous build and test systems significantly increase development and maintenance costs and slow/delay integrations.
  - Homogenous build and test systems across teams and software reduce development and maintenance costs and speeds integrations. (i.e. **CASL**)

One of biggest impediments to improving development and integration workflows is developer inability/unwillingness to learn git!