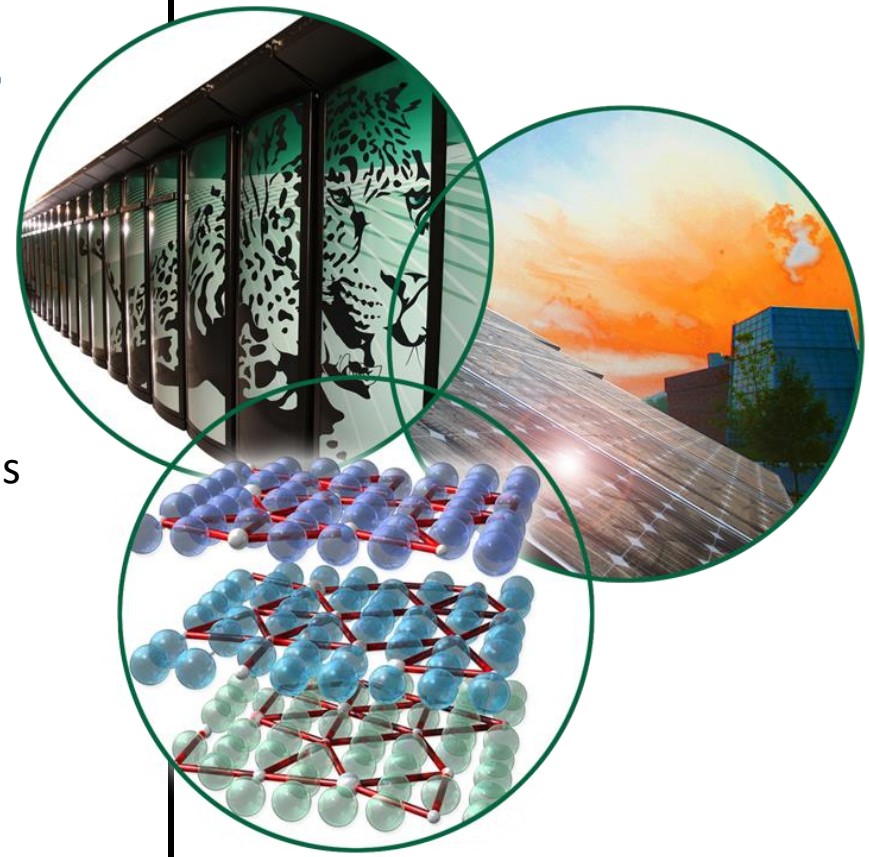


Software Engineering Processes for the CASL Innovation Hub

Roscoe A. Bartlett, Ph.D.

bartlettra@ornl.gov

http://web.ornl.gov/~8vt/

Computational Engineering and Energy Sciences Group,
Oak Ridge National Laboratory

Center for Computing Research Seminar
Sandia National Laboratories
October 26, 2015

U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

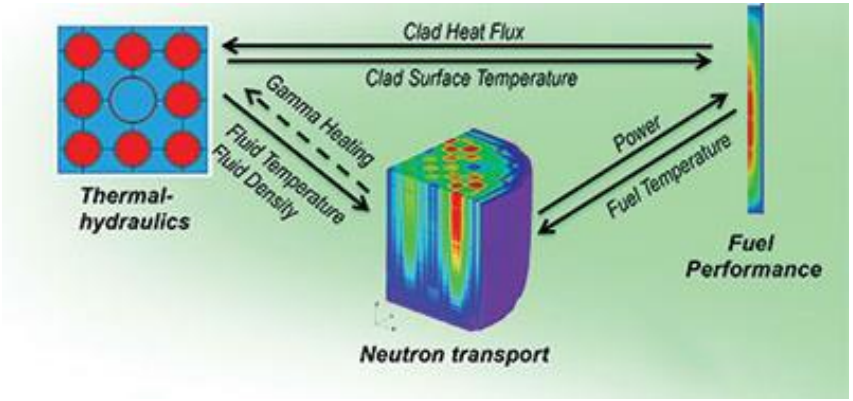
Overview of CASL

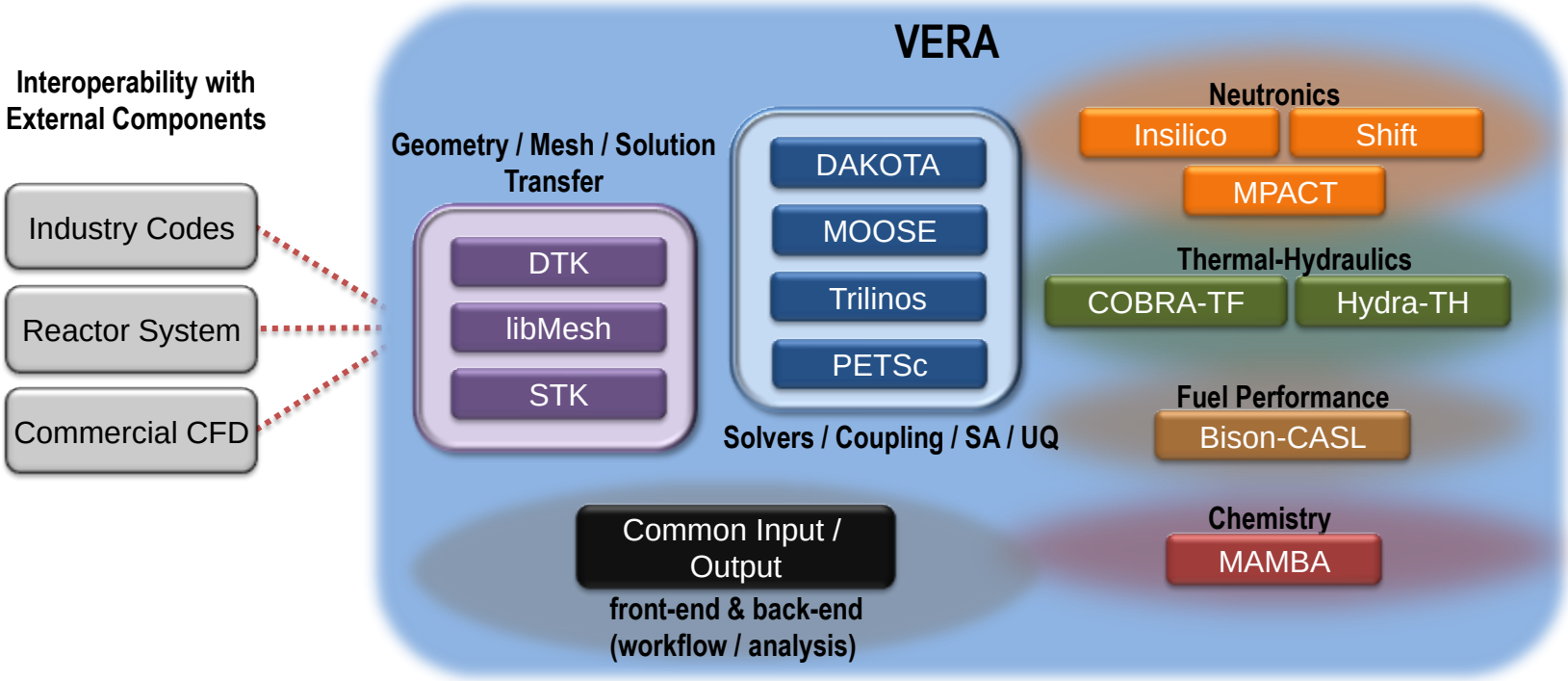
Overview of CASL

- **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 light-water nuclear reactors
  - Produce a set of simulation tools to model light-water 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), Fuel Materials & Chemistry (FMC), Validation & Modeling Applications (VMA), Technology Development & Outreach (TDO)

CASL VERA Core Simulator

Focus: Modeling of the Reactor Core for Nuclear Light-water Reactors



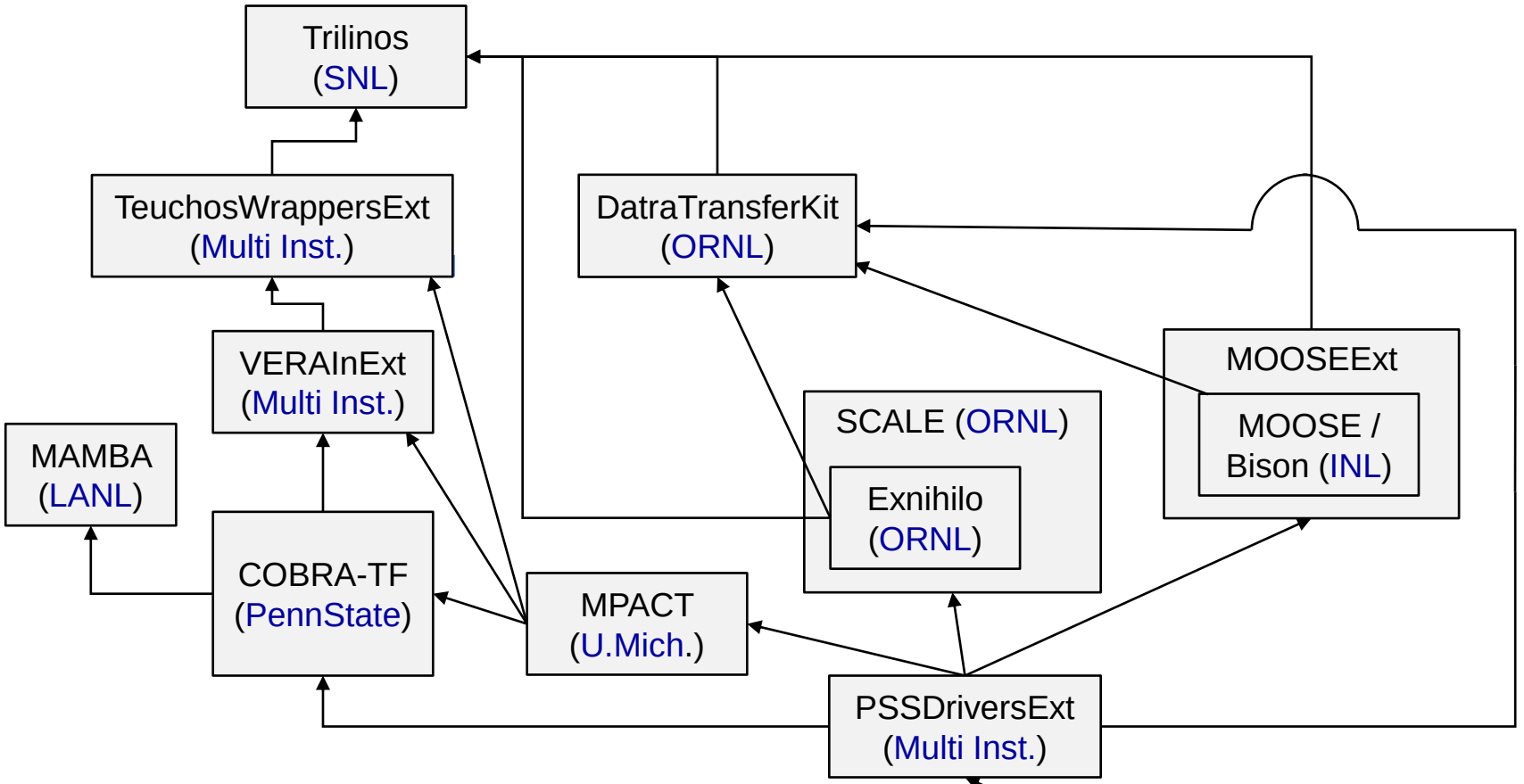


Overview of CASL VERA Development

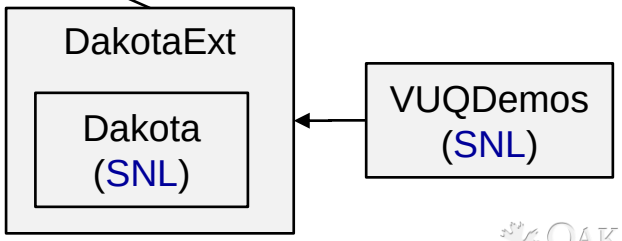
CASL VERA Development Overview

- VERA Development is complicated in almost every way ☹
- VERA Currently Composed of:
  - 19 different git repositories on casl-dev.ornl.gov (clones of other repos) most with a different access list (NDAs, Export Control, IP, etc.)
  - Integrating development efforts from many teams from 9+ institutions
- CMake build system using TriBITS CMake Framework:
  - Over 2700 CMakeLists.txt files!
- VERA Software Development Process:
  - Official definition of VERA is 'master' branch of git repos under gitolite control at git@casl-dev.ornl.gov:<repo-name>.
  - Primary development platform: CASL Fissile/Spy Machines
  - VERA integration maintained by continuous and nightly testing:
    - Pre-push CI testing: checkin-test-vera.sh, cloned VERA git repos, on Fissile machine.
    - Post-push CI testing: CTest/CDash, all VERA git repos, shared libs.
    - Nightly testing: MPI and Serial builds, Debug and Release builds, …
    - 100% passing builds and tests!
  - VERA snapshots and releases are taken off of 'master' branches on casl-dev git repos.

Dependencies Between Selected VERA Repositories



- Primary/originating institution shown in Blue
- Most codes being contributed by multiple institutions as well
- All direct dependencies **not** shown
- Dependencies between repos are though TriBITS package dependencies
- Local VERA git clones of all these repos kept compatible



TriBITS Structural Units and VERA Software Organization

What is TriBITS?

- Framework for large, distributed multi-repository CMake projects
- Reduce boiler-plate CMake code and enforce consistency across large distributed projects
- Subproject dependencies and namespacing architecture (packages)
- Automatic package dependency handling (directed acyclic graph)
- Additional functionality missing in raw CMake
- Change default CMake behavior when necessary
- Additional tools for agile software development processes (e.g. Continuous Integration (CI))

History of TriBITS:

- 2007: Initially developed as a CMake package architecture for Trilinos
- 2011: Generalized and extended for CASL VERA
- 2014: Source code hosted on GitHub

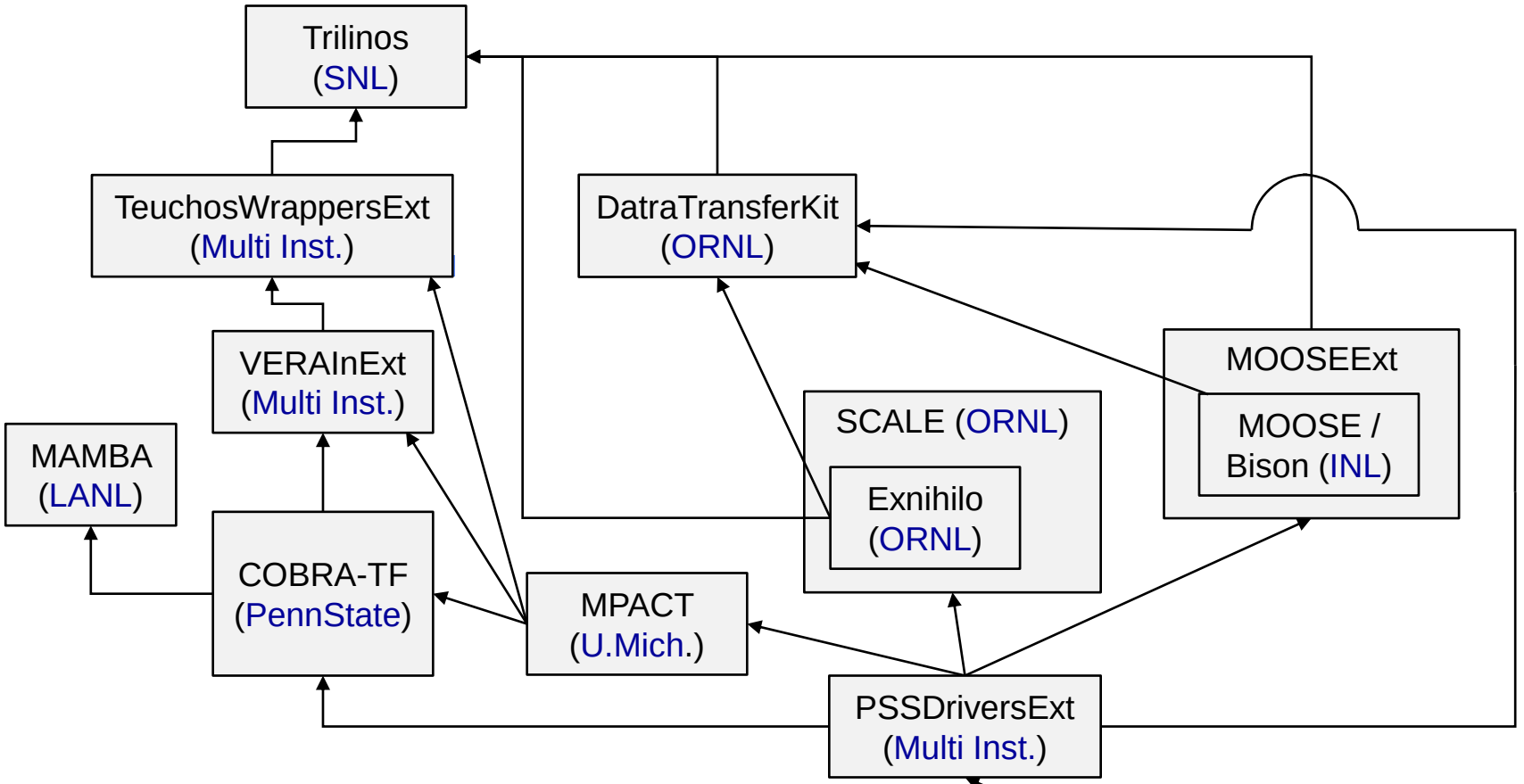
TriBITS Structural Units

- TriBITS Project:
  - Complete CMake “Project”
  - Overall projects settings
- TriBITS Repository:
  - Collection of Packages and TPLs
  - Unit of distribution and integration
  - Typically a version control (git) repository
- TriBITS Package:
  - Encapsulated collection of related software & tests
  - Unit of testing, namespacing, documentation, and reuse
  - Lists dependencies on SE Packages & TPLs
- TriBITS Subpackage:
  - Optional partitioning of package software & tests
  - Primarily for dependency management (SE principles)
- TriBITS TPLs (Third Party Libraries):
  - Specification of external dependencies (libs)
  - Required or optional dependency
  - Single definition across all packages
  - Can use native CMake Find<Package>.cmake modules

Packages + Subpackages = Software Engineering (SE) Packages

See: https://tribits.org/doc/TribitsDevelopersGuide.html

TriBITS and VC Repos for CASL VERA



- Outer boxes are all TriBITS and VC repos (all native git or snapshotted to git)
- Inner boxes (e.g. Exnihilo, MOOSE, Dakota) are VC repos but not TriBITS repos

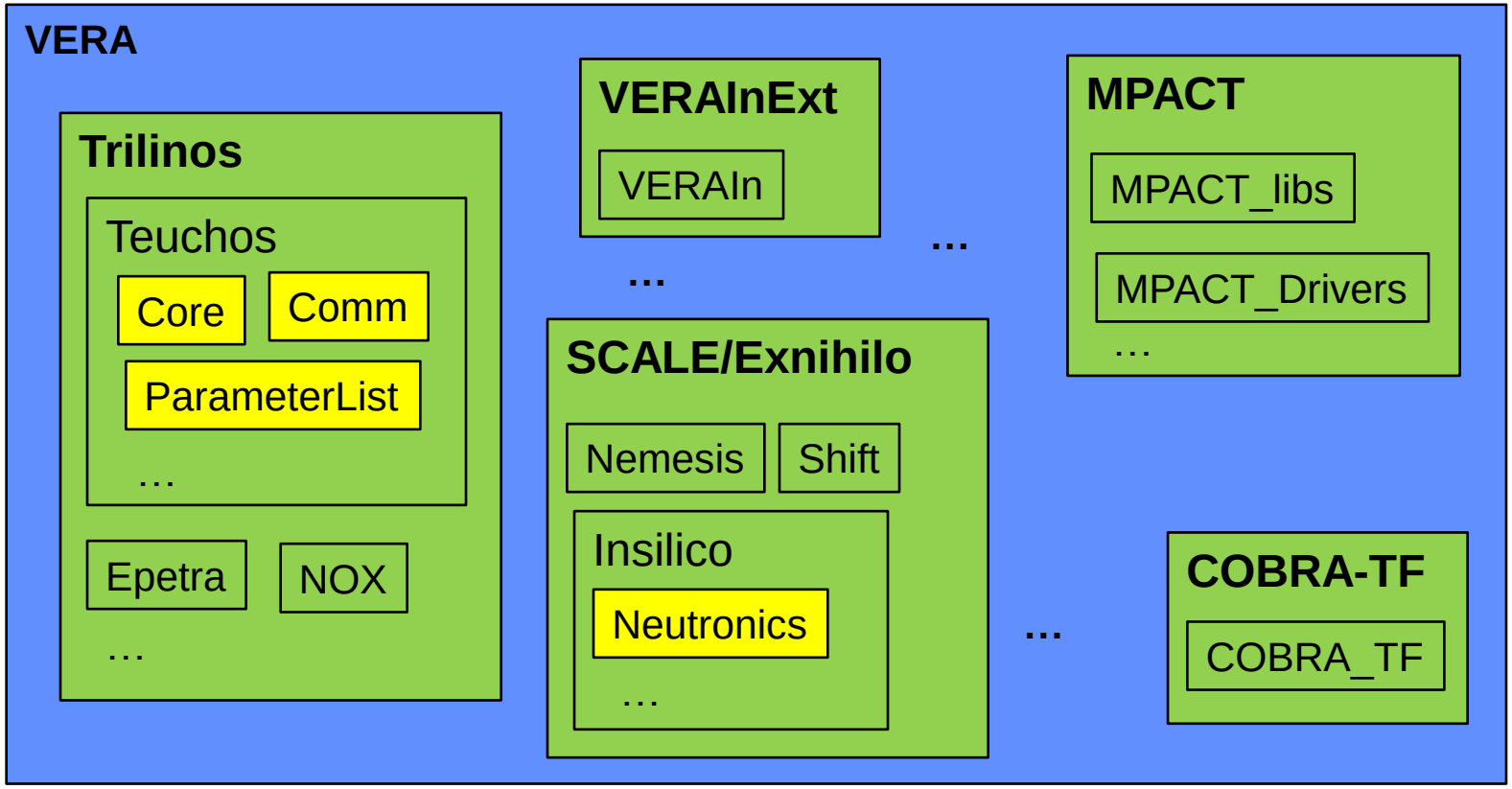
VERA/cmake/ExtraRepositoriesList.cmake

```
TRIBITS_PROJECT_DEFINE_EXTRA_REPOSITORIES(
  TriBITS             ""   GIT  git@casl-dev:TriBITS          ""  ${TriBITS_REPO_TYPE}
  Trilinos            ""   GIT  git@casl-dev:Trilinos         ""  Continuous
  TeuchosWrappersExt "" GIT git@casl-dev:TeuchosWrappersExt "" Continuous
  MAMBA               ""   GIT  git@casl-dev:MAMBA            ""  Continuous
  COBRA-TF            ""   GIT  git@casl-dev:COBRA-TF         ""  Continuous
  VERAInExt           ""   GIT  git@casl-dev:VERAInExt        ""  Continuous
  VERAData            ""   GIT  git@casl-dev:VERAData         NOPACKAGES ${VERADATA_CAT}
  DataTransferKit ""   GIT git@casl-dev:DataTransferKit "" Continuous
  MOOSEExt            ""   GIT  git@casl-dev:MOOSEExt         ""  Continuous
  MOOSE             MOOSEExt/MOOSE   GIT
      git@casl-dev:MOOSE   NOPACKAGES  Continuous
  SCALE               ""   GIT  git@casl-dev:SCALE           ""  Continuous
  Exnihilo          SCALE/Exnihilo   GIT
      git@casl-dev:Exnihilo                        NOPACKAGES  Continuous
  MPACT               ""   GIT  git@casl-dev:MPACT           ""  Continuous
  LIMEExt             ""   GIT  git@casl-dev:LIMEExt         ""  Nightly
  PSSDriversExt    ""   GIT  git@casl-dev:PSSDriversExt  ""  Continuous
  DakotaExt       ""   GIT  git@casl-dev:DakotaExt ""  Continuous
  Dakota  DakotaExt/Dakota  GIT  git@casl-dev:Dakota  NOPACKAGES  Continuous
  VUQDemos            ""   GIT  git@casl-dev:VUQDemos         ""  Nightly
)
```

Official version of VERA in on master branch used for CI & Nightly testing

- Partial set of repos can be cloned (protected by different groups)
- Non-git repos are converted into git repos: Dakota (svn), SCALE (hg), MOOSE (git submodules)

VERA Meta-Project, Repositories, Packages & Subpackages

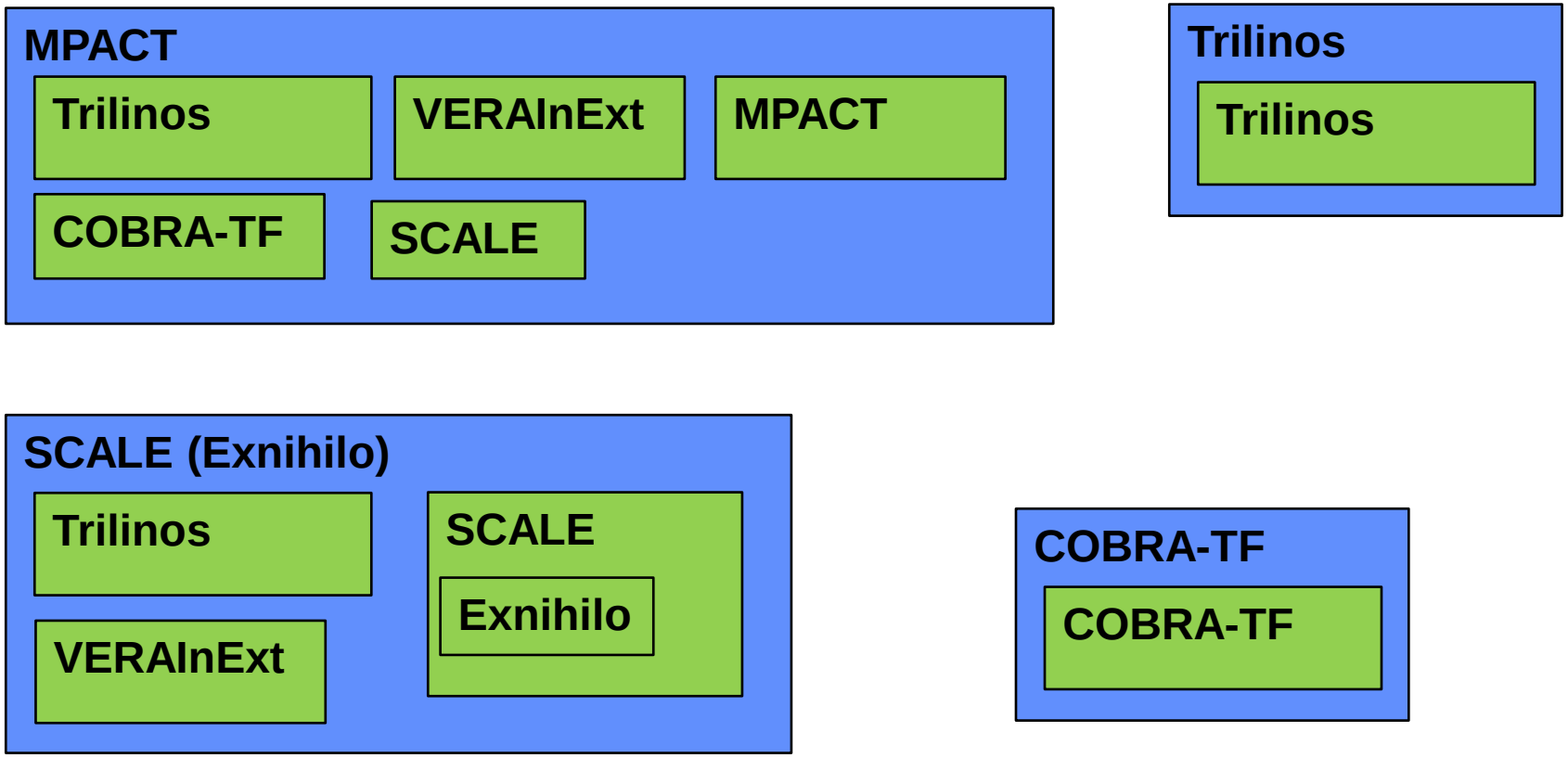


- **VERA meta-project:** Git repository and TriBITS meta-project (contains no packages)
- **TriBITS repos:**: Trilinos, VERAInExt, COBRA-TF, MPACT, SCALE …
- **TriBITS packages**: Teuchos, Epetra, VERAIn, Insilico, COBRA_TF, MPACT_Drivers, …
- **TriBITS subpackages**: TeuchosCore, InsilicoNeutronics …
- **TriBITS SE packages**: TeuchosCore , Teuchos, VERAIn, Insilico, InsilicNeutronics, …

Current Adoption and Usage of TriBITS in CASL

- Repositories of independent projects using **TriBITS primary build/test system**:
  - Trilinos: SNL
  - SCALE: ORNL
    - Requires GCC 4.6.1+ and Intel 13.1+
    - Mixed Fortran, C, C++
    - Linux builds
    - Windows builds
  - Exnihilo: ORNL
    - Mostly C++ with some Fortran 90/77, Python, etc.
    - Contains Denovo, Shift, Insilico
  - MPACT: Univ. of Mich.
    - Requires GCC 4.6.1+ and Intel 13.1+
    - Mostly Fortran
    - Windows builds
  - COBRA-TF: Penn. State Univ.
    - Mostly Fortran 77 and 90
- Native TriBITS repos providing packages: TeuchosWrappersExt, VERAInExt, DataTransferKit
- VERA Repositories/packages not using TriBITS as native build system but have **secondary native TriBITS support**: MAMBA
- VERA Repositories not providing a TriBITS build: MOOSE, DAKOTA

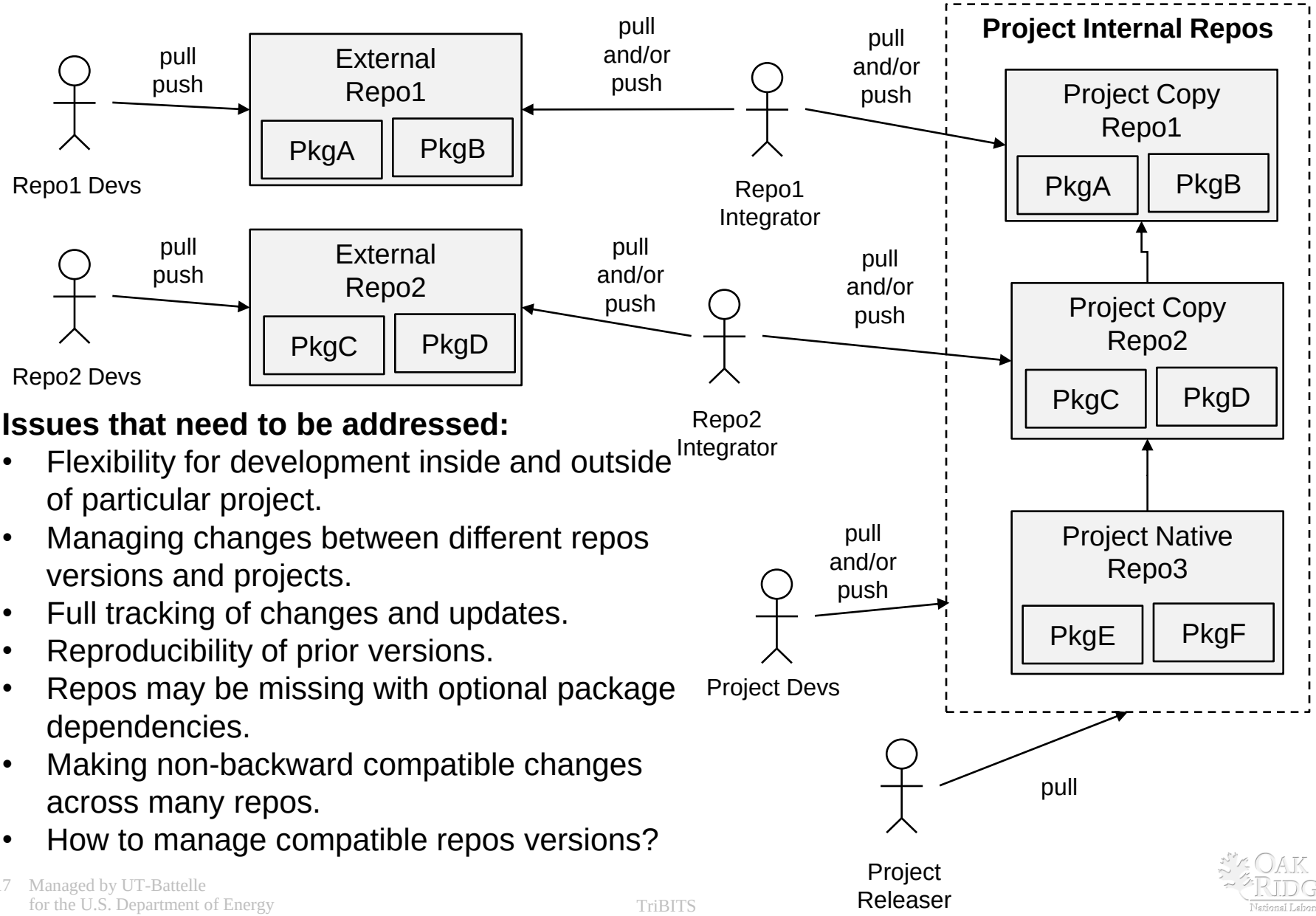
Flexibility in TriBITS Projects and Repositories



The same TriBITS repositories can be arranged into multiple TriBITS CMake projects!

Multi-Repository Support and Handling

Managing Compatible Repos and Repo Versions



Issues that need to be addressed:

- Flexibility for development inside and outside of particular project.
- Managing changes between different repos versions and projects.
- Full tracking of changes and updates.
- Reproducibility of prior versions.
- Repos may be missing with optional package dependencies.
- Making non-backward compatible changes across many repos.
- How to manage compatible repos versions?

Managing Multiple Compatible Git Repo Versions?

- **Snapshot all repos into one big repo** (e.g. SIERRA/Trilinos style)?
  - Advantages:
    - One set of SHA1s, easy to do git bisect
    - One repo to pull and build final version
    - Can pull in non-git repos (e.g. Hg, svn)
  - Disadvantages:
    - Harder to coordinate changes back and forth to native repos
    - Does not allow for partitioning based on access control
- **Use git modules?**
  - Advantages:
    - Built-in git support and documentation
    - Individual repos stay independent (let git do its job).
    - Tracks compatible versions of repos, allows basic git bisect
  - Disadvantages:
    - Extra commands to pull component repos
    - Updating repos versions is complex for non-git savvy developers
    - Does not support co-development well at all
- **Treat set of repos as one big repo** (e.g. SIERRA `code`, `test` & `doc`) (**gitdist**)
  - Advantage: Simple single-repo workflow
  - Disadvantage: Must track compatible versions as add-on (e.g. **<Project>RepoVersion.txt** files)

CASL VERA => Uses gitdist for most git repos, uses snapshotting for non-native or more complex git repos.

gitdist: Treat collection of git repos as one

- **gitdist**: Simple stand-alone Python tool for distributing git commands across multiple git repos.

```
Usage: gitdist [gitdist options] <raw-git-command> [git options]
```

- .gitdist.default file committed to base git repo:

```
Trilinos
SCALE
SCALE/Exnihilo
…
```

Example:

```
$ gitdist status
*** Base Git Repo: VERA
On branch master
…
*** Git Repo: Trilinos
On branch master
…
*** Git Repo: SCALE
# On branch master
…
*** Git Repo: SCALE/Exnihilo
# On branch master
…
```

- Same workflow as single git repo:

```
$ gitdist checkout master
$ gitdist pull
<create commits to repos>
$ gitdist status
$ gitdist push
```

- No complexities of git submodules
- Keeps full history of separate repos for easy merging, etc. (unlike snapshots)
- Special commands for many repos:
  - Show compat status table:

    ```
    $ gitdist-status
    ```

  - Show only changed repos:

    ```
    $ gitdist-mod-status
    $ gitdist-mod local-stat
    ```

gitdist: Show summary status table

```
$ gidist-status   # alias 'gitdist dist-repo-status'
```

ID	Repo Dir	Branch	Tracking Branch	C	M	?
0	VERA (Base)	master	origin/master	2	1	
1	TriBITS	master	origin/master			
2	Trilinos	master	origin/master			
3	TeuchosWrappersExt	master	origin/master			2
4	MAMBA	master	origin/master			
5	COBRA-TF	master	origin/master			
6	VERAInExt	master	origin/master			3
7	DataTransferKit	master	origin/master			
8	MOOSEExt	master	origin/master			
9	MOOSEExt/MOOSE	master	origin/master			
10	SCALE	master	origin/master			
11	SCALE/Exnihilo	master	origin/master			
12	MPACT	master	origin/master	2		
13	LIMEExt	master	origin/master			
14	hydrath	master	origin/master			
15	PSSDriversExt	master	origin/master		4	
16	DakotaExt	master	origin/master			
17	DakotaExt/Dakota	master	origin/master			
18	VUQDemos	master	origin/master			

gitdist: Show summary status table, modified only

```
$ gidist-mod-status # alias 'gitdist dist-repo-status --dist-mod-only'

-------------------------------------------------------------------
| ID | Repo Dir            | Branch | Tracking Branch | C | M | ? |
|----|---------------------|--------|-----------------|---|---|---|
|  0 | VERA (Base)         | master | origin/master   | 2 | 1 |   |
|  3 | TeuchosWrappersExt  | master | origin/master   |   |   | 2 |
|  6 | VERAInExt           | master | origin/master   |   |   | 3 |
| 12 | MPACT               | master | origin/master   | 2 |   |   |
| 15 | PSSDriversExt       | master | origin/master   |   | 4 |   |
-------------------------------------------------------------------
```

- Compress out repos with no changes
- Manage changes on many repos at once even when there are 100s of repos!

gitdist: Run commands only on modified repos

```
$ gitdist-mod log --oneline --name-status HEAD ^@{u}

*** Base Git Repo: VERA
0ed6639 Minor comment update
M       cmake/ctest/drivers/fissile4/load_dev_env_gcc-4.8.3.sh


*** Git Repo: TriBITS
db7152a Add better gitdist-mod-status command
M       test/python_utils/gitdist_UnitTests.py
M       tribits/devtools_install/load_dev_env.csh.in
M       tribits/devtools_install/load_dev_env.sh.in
M       tribits/python_utils/gitdist
```

- See detailed changes only in changed repos
- Manage changes on many repos at once even when there are 100s of repos!

TriBITS: clone_extra_repos.py

```
$ ./clone_extra_repos.py
…
```

ID	Repo Name	Repo Dir	VC	Repo URL	Category
1	TriBITS	TriBITS	GIT	git@casl-dev:TriBITS	Continuous
2	Trilinos	Trilinos	GIT	git@casl-dev:Trilinos	Continuous
3	TeuchosWrappersExt	TeuchosWrappersExt	GIT	git@casl-dev:TeuchosWrappersExt	Continuous
4	MAMBA	MAMBA	GIT	git@casl-dev:MAMBA	Continuous
5	COBRA-TF	COBRA-TF	GIT	git@casl-dev:COBRA-TF	Continuous
6	VERAInExt	VERAInExt	GIT	git@casl-dev:VERAInExt	Continuous
7	DataTransferKit	DataTransferKit	GIT	git@casl-dev:DataTransferKit	Continuous
8	MOOSEExt	MOOSEExt	GIT	git@casl-dev:MOOSEExt	Continuous
9	MOOSE	MOOSEExt/MOOSE	GIT	git@casl-dev:MOOSE	Continuous
10	SCALE	SCALE	GIT	git@casl-dev:SCALE	Continuous
11	Exnihilo	SCALE/Exnihilo	GIT	git@casl-dev:Exnihilo	Continuous
12	MPACT	MPACT	GIT	git@casl-dev:MPACT	Continuous
13	LIMEExt	LIMEExt	GIT	git@casl-dev:LIMEExt	Continuous
14	hydrath	hydrath	GIT	git@casl-dev:hydrath	Nightly
15	PSSDriversExt	PSSDriversExt	GIT	git@casl-dev:PSSDriversExt	Continuous
16	DakotaExt	DakotaExt	GIT	git@casl-dev:DakotaExt	Continuous
17	Dakota	DakotaExt/Dakota	GIT	git@casl-dev:Dakota	Continuous
18	VUQDemos	VUQDemos	GIT	git@casl-dev:VUQDemos	Nightly

```
…
Running: git clone git@casl-dev:TriBITS TriBITS

Running: git clone git@casl-dev:Trilinos Trilinos
…
```

- Reads from ExtraRepositoriesList.cmake
- Only clones the repos that the user/developer has access to clone!

TriBITS: Keeping track of compatible sets of repos

How to keep track of compatible sets of repos?

=> TriBITS support for <Project>RepoVersion.txt file:

```
*** Base Git Repo: SomeBaseRepo
e102e27 [Mon Sep 23 11:34:59 2013 -0400] <author1@someurl.com>
First summary message
*** Git Repo: ExtraRepo1
b894b9c [Fri Aug 30 09:55:07 2013 -0400] <author2@someurl.com>
Second summary message
*** Git Repo: ExtraRepo2
97cf1ac [Thu Dec 1 23:34:06 2011 -0500] <author3@someurl.com>
Third summary message
...
```

Setting -D<PROJECT>_GENERATE_REPO_VERSION_FILE=ON results in <Project>RepoVersion.txt getting:

- generated in the build base directory,
- echoed in the configure output (therefore archived to CDash),
- installed in the base install directory,
- included in the source tarball ('make package_source'),
- installed in the base install directory from the untarred source.

Use with gitdist to access compatible versions:

```
$ gitdist --dist-repo-file=<Project>RepoVersion.<somedate>.txt [other options]
```

Using VERARepoVersion.txt for Dev Distributions

- Known “good” versions of the Project code are recorded as VERARepoVersion.txt files (e.g. archived on CDash, committed directly to base repo, etc.).
- **Example:** If a given Nightly build of Project passed on all platforms then we can give the associated VERARepoVersion.txt file as the “version” for a client to use.
- Send client file VERARepoVersion.<newdate>.txt for “good” version to install.
- Client gets updated version:

```
$ cd VERA/
$ gitdist fetch
$ gitdist --dist-version-file=~/VERARepoVersion.<newdate>.txt \
  checkout _VERSION_
```

- Client can see changes since a previous installs of <Project>:

```
$ gitdist fetch
$ gitdist \
  --dist-version-file=~/VERARepoVersion.<newdate>.txt \
  --dist-version-file2=${INSTALL_BASE}/<olddate>/VERARepoVersion.txt \
  log-short --name-status _VERSION_ ^_VERSION2_
```

- NOTE: <Project>RepoVersion.txt could be committed to the base repo (at well-defined points) to allow using git bisect to search for introduction of defects and other changes!

Pre-Push CI Testing and Pushing of Multiple Repos

$ cat checkin-test-vera.sh

```
#!/bin/bash -e
…
$VERA_BASE_DIR/VERA/checkin-test.py \
--src-dir=$VERA_BASE_DIR_ABS/VERA \
--extra-repos-file=project \
--extra-repos-type=Continuous \
--ignore-missing-extra-repos \
--default-builds=MPI_RELEASE_DEBUG_SHARED \
-j16 \
--ctest-timeout=400 \
$EXTRA_ARGS
```

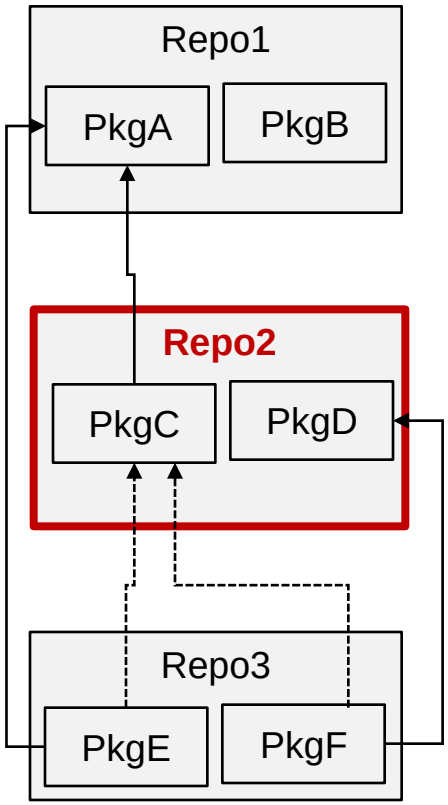
- Very thin bash script wrapper checkin-test-vera.sh for TriBITS checkin-test.py
- Automatically picks up cloned repos listed in ExtraRepositoriesList.cmake
- Safe pushes requires all affected repos to be cloned and available

Dealing with Missing Repos giving Missing Packages

What if **Repo2** is missing? Can we still configure and build the remaining packages in Repo3?

```
# Repo3/PkgE/cmake/Dependencies.cmake
   LIB_REQUIRED_PACKAGES PkgA
   LIB_OPTIONAL_PACKAGES PkgC
…
TRIBITS_ALLOW_MISSING_EXTERNAL_PACKAGES(PkgC)

# Repo3/PkgF/cmake/Dependencies.cmake
  LIB_REQUIRED_PACKAGES  PkgD
  LIB_OPTIONAL_PACKAGES  PkgC
…
TRIBITS_ALLOW_MISSING_EXTERNAL_PACKAGES(PkgC  PkgD)
```



Now when configuring the Project with Repo2 missing TriBITS automatically adjusts:

```
NOTE: PkgC is being ignored since its directory is missing and
PkgC_ALLOW_MISSING_EXTERNAL_PACKAGE = TRUE!
…
NOTE: Setting <PROJECT>_ENABLE_PkgF=OFF because PkgD is a
required missing package!
```

Primary Meta-Project Packages (PMPP)

- Some packages are “primary” to the project and are under development by the project. Other packages are just there to satisfy downstream dependencies.

Example: VERA

```
SET(Trilinos_NO_PRIMARY_META_PROJECT_PACKAGES TRUE)
SET(SCALE_NO_PRIMARY_META_PROJECT_PACKAGES TRUE)
SET(SCALE_NO_PRIMARY_META_PROJECT_PACKAGES_EXCEPT Insilico)
SET(LIMEExt_NO_PRIMARY_META_PROJECT_PACKAGES TRUE)
```

- -DVERA_ENABLE_ALL_PACKAGES=ON: Only explicitly enable the PMPPs for VERA and not packages in Trilinos, SCALE (except Insilico), LIME, etc.

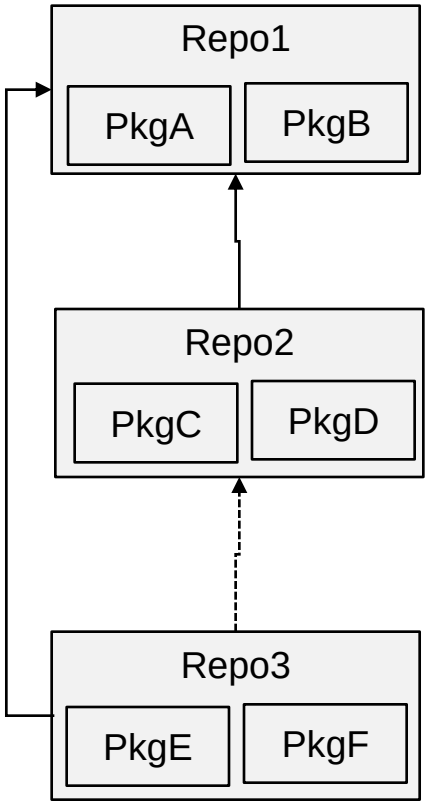
- -DVERA_ENABLE_ALL_TESTS=ON: Only enable tests for PMPPs that are explicitly enabled and not others that might be enabled in Trilinos, SCALE, etc.

- Used in:
  - Automated CTest/CDash testing (only process PMPPs)
  - Pre-push CI testing with checkin-test.py (tests for only PMPPs)
  - Creating source tarball distributions (excludes packages not enabled)

Incorporating Externally Configured/Built Software

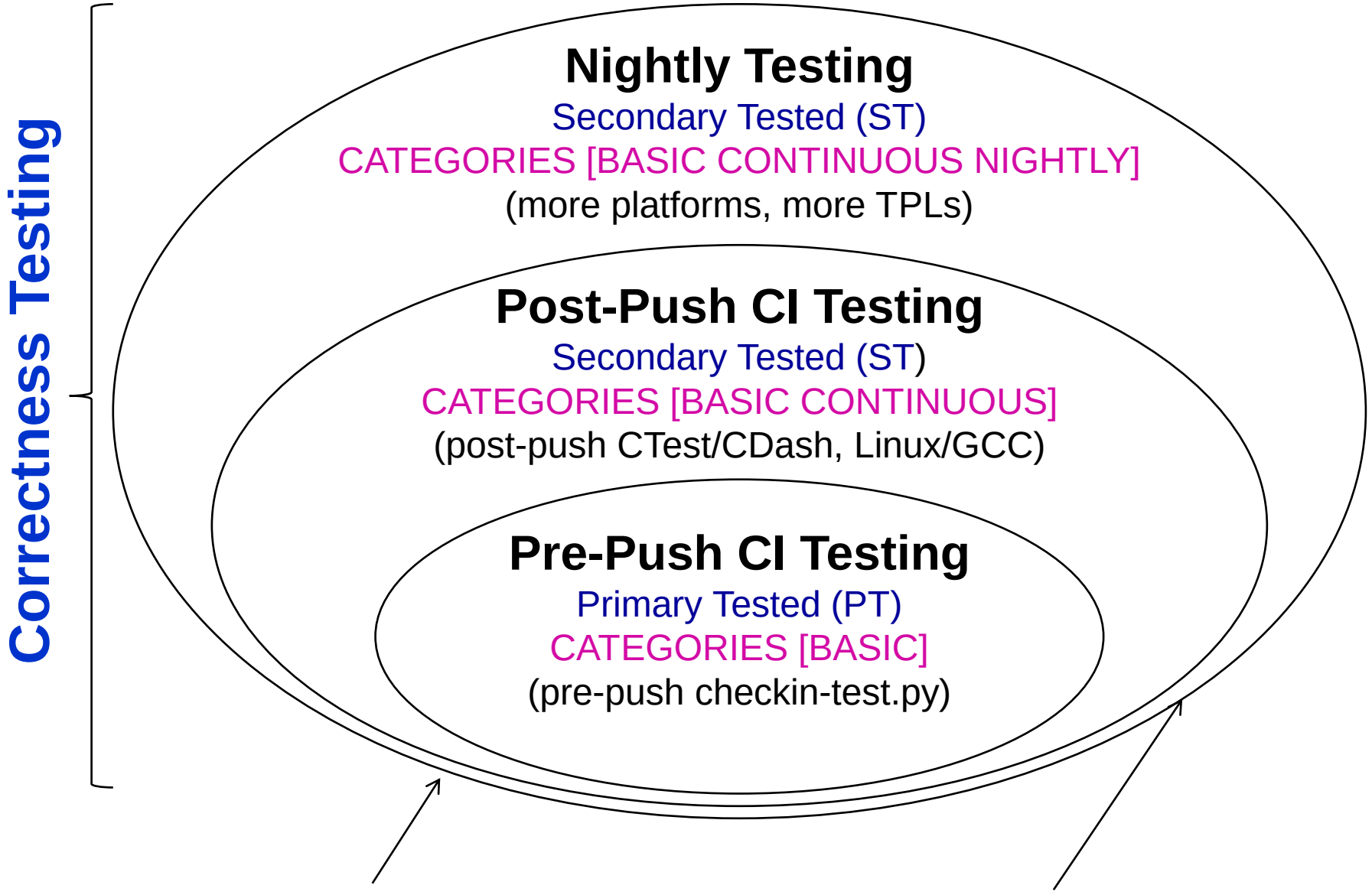
- **Motivation**: For some software, it may not be practical or maintainable to create a (secondary) native TriBITS build for a piece of software.
- **Goal:**
  - Use another configure/build tool for external software.
  - Put in CMake/TriBITS hooks to incorporate into TriBITS build with other packages.
- **Easy case:** No upstream TriBITS packages => ***Repo1***
- **Medium case:** No downstream TriBITS packages => ***Repo3***
- **Hard case:** Both upstream and downstream TriBITS packages => ***Repo2***
- **Example:** INL MOOSE developers not willing to support a native TriBITS build of libmesh and MOOSE/Bison for CASL VERA. Libmesh depends on DataTransferKit and therefore Trilinos ☹

  Tpetra <= DataTransferKit <= libmesh <= MOOSE <= Tiamat
- **Technical challenges in TriBITS:**
  - Generate export makefile for upstream Trilinos packages
  - Create CMake rules to produce libs/executables
  - Add dependencies for changes to upstream code
  - Add dependencies for modified external project files



Testing

TriBITS Standard Testing Layers



Pre-Push CI Testing: checkin-test.py

```
$ checkin-test.py --do-all –push
```

- Integrates with latest version in remote git repositories
- Figures out modified packages

```
Modified file: 'packages/teuchos/CMakeLists.txt'
  => Enabling 'Teuchos'!
```

- Enables all forward/downstream packages & tests
- Configures, builds, and runs tests
- Does the push (if all builds/tests pass)
- Sends notification emails
- Fully customizable (enabled packages, build cases, etc.)
- Documentation: checkin-test.py --help

Post-Push Testing: TRIBITS_CTEST_DRIVER()


My CDash All Dashboards Log Out — Monday, May 05 2014 20:48:48 EDT

VERA

Dashboard Calendar Previous Current Next Project Settings

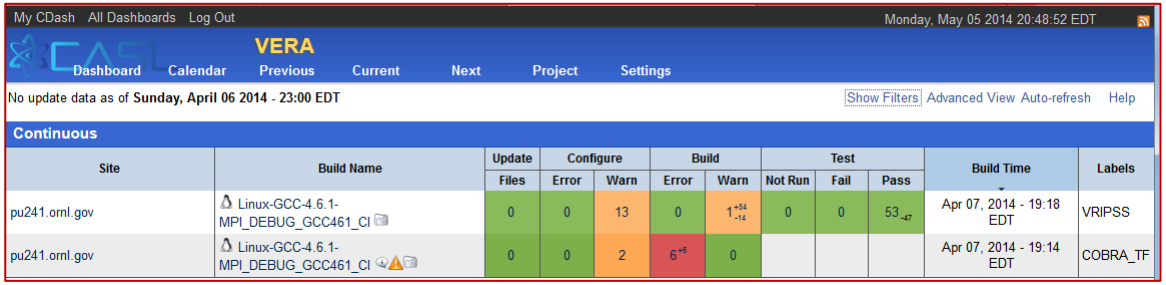
No update data as of **Sunday, April 06 2014 - 23:00 EDT** — Show Filters Advanced View Auto-refresh Help

Nightly

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_RELEASE_GCC461	0	0	74	0	60	0	1	625	Apr 07, 2014 - 01:11 EDT	(13 labels)
james007.ornl.gov	Linux-GCC-4.6.1-MPI_RELEASE_GCC461_WEEKLY	0	0	74	0	60	0	1	634	Apr 07, 2014 - 01:15 EDT	(13 labels)
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461	0	0	74	0	61	0	0	626	Apr 07, 2014 - 01:11 EDT	(13 labels)

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	15	0	1	0	0	320	Apr 07, 2014 - 21:38 EDT	(2 labels)
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	15	6	1	0	0	53	Apr 07, 2014 - 19:14 EDT	(2 labels)
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	44	0	120	0	0	594	Apr 07, 2014 - 04:06 EDT	(11 labels)


My CDash All Dashboards Log Out — Monday, May 05 2014 20:48:52 EDT

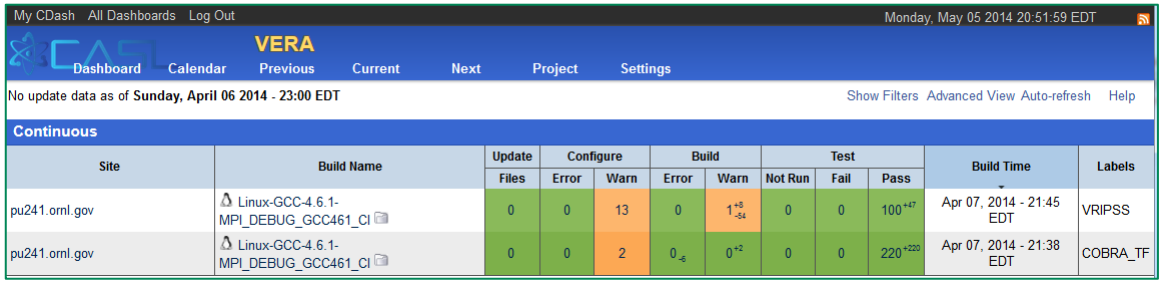
VERA

Dashboard Calendar Previous Current Next Project Settings

No update data as of **Sunday, April 06 2014 - 23:00 EDT** — Show Filters Advanced View Auto-refresh Help

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	13	0	1^{+54}_{-14}	0	0	53_{-47}	Apr 07, 2014 - 19:18 EDT	VRIPSS
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	2	6^{+6}	0				Apr 07, 2014 - 19:14 EDT	COBRA_TF


My CDash All Dashboards Log Out — Monday, May 05 2014 20:51:59 EDT

VERA

Dashboard Calendar Previous Current Next Project Settings

No update data as of **Sunday, April 06 2014 - 23:00 EDT** — Show Filters Advanced View Auto-refresh Help

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	13	0	1^{+8}_{-54}	0	0	100^{+47}	Apr 07, 2014 - 21:45 EDT	VRIPSS
pu241.ornl.gov	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	2	0_{-6}	0^{+2}	0	0	220^{+220}	Apr 07, 2014 - 21:38 EDT	COBRA_TF

VERA CDash Dashboard for 4/6/2014

- Collapsed summaries for each build case
- Nightly, CI, Experimental build cases

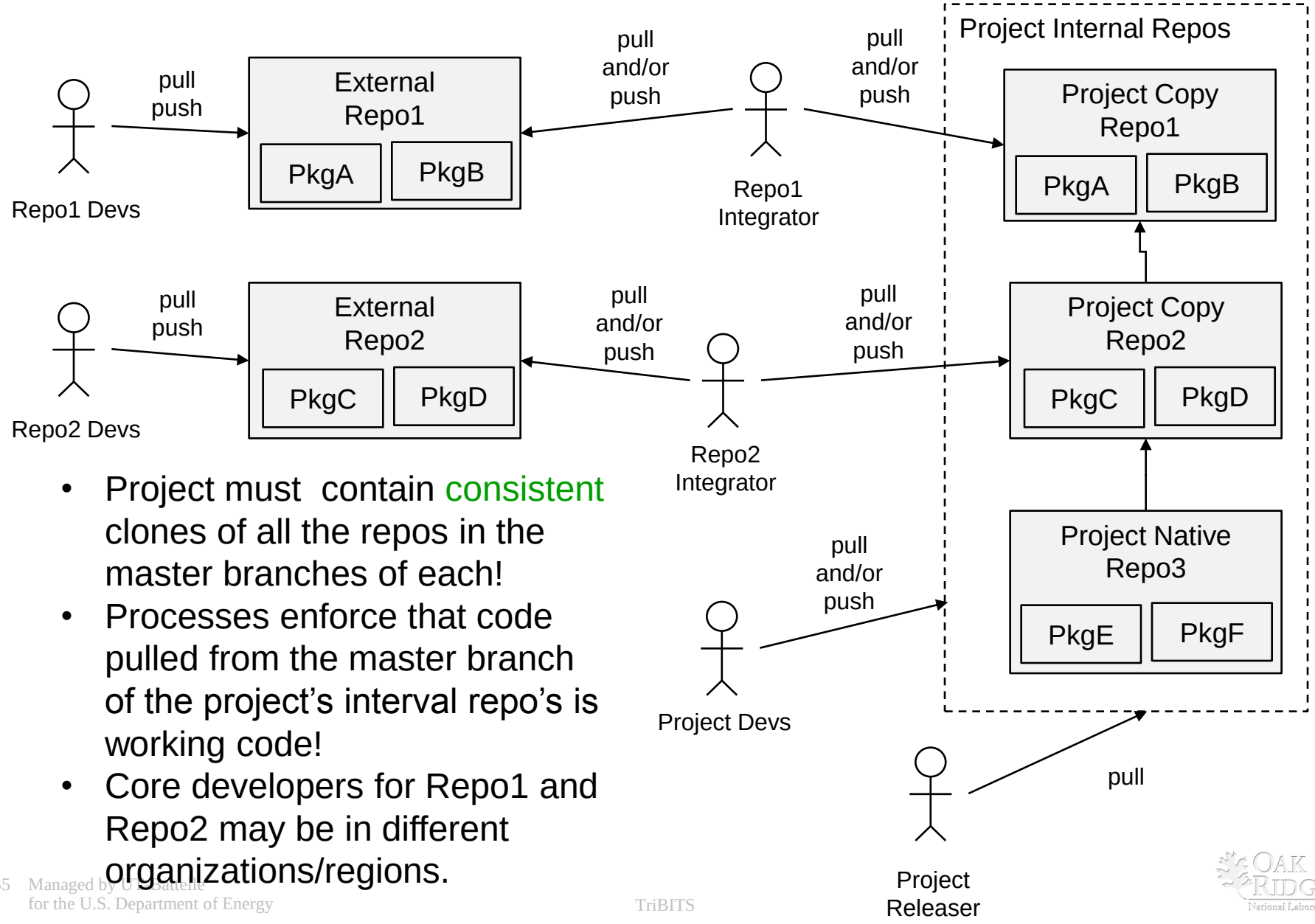
VERA CDash CI Iterations

- Individual packages built in sequence
- Targeted emails for failed package build & tests
- Failed packages disabled in downstream packages

  => Don't propagate failures!

Multi-Repository Integration Models and Processes

Integrating Repos into Project: External and Internal



- Project must contain consistent clones of all the repos in the master branches of each!
- Processes enforce that code pulled from the master branch of the project's interval repo's is working code!
- Core developers for Repo1 and Repo2 may be in different organizations/regions.

Multi-Repository Integration Models

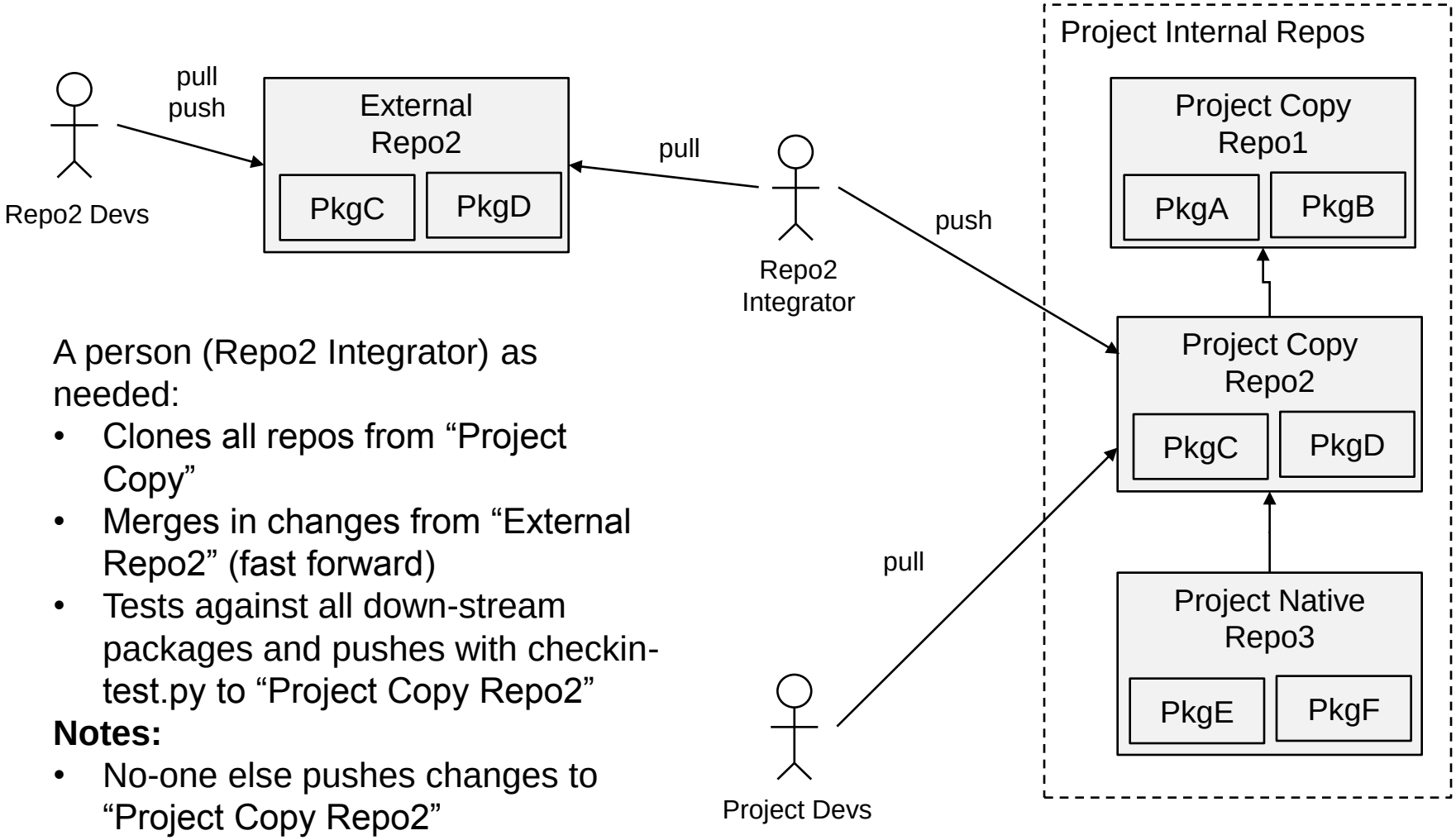
- Range of development and sync models (external dev to internal dev):
  - External repo is manually synced into project/master as needed.
  - External repo is synced automatically using sync server into project/master using the checkin-test.py script.
  - Both external and internal repos pushed to by different development groups with sync servers running one way or both ways.
  - Internally managed repo is synced to an external repo on some schedule to make available to other developers and users and changes from external repo may or may not be synced back into internal repo.
  - Internally managed repo

- A given repo may shift between different integration models at different periods of time (e.g. Trilinos, COBRA-TF)

- Integration of different repos should be done independently if possible (e.g. errors in MPACT should not stop pushes of SCALE/Exnihilo and visa versa).

- Non-backward compatible changes to upstream repos require coordinated development and combined pushing to project/master

External Repo is manually synced



A person (Repo2 Integrator) as needed:

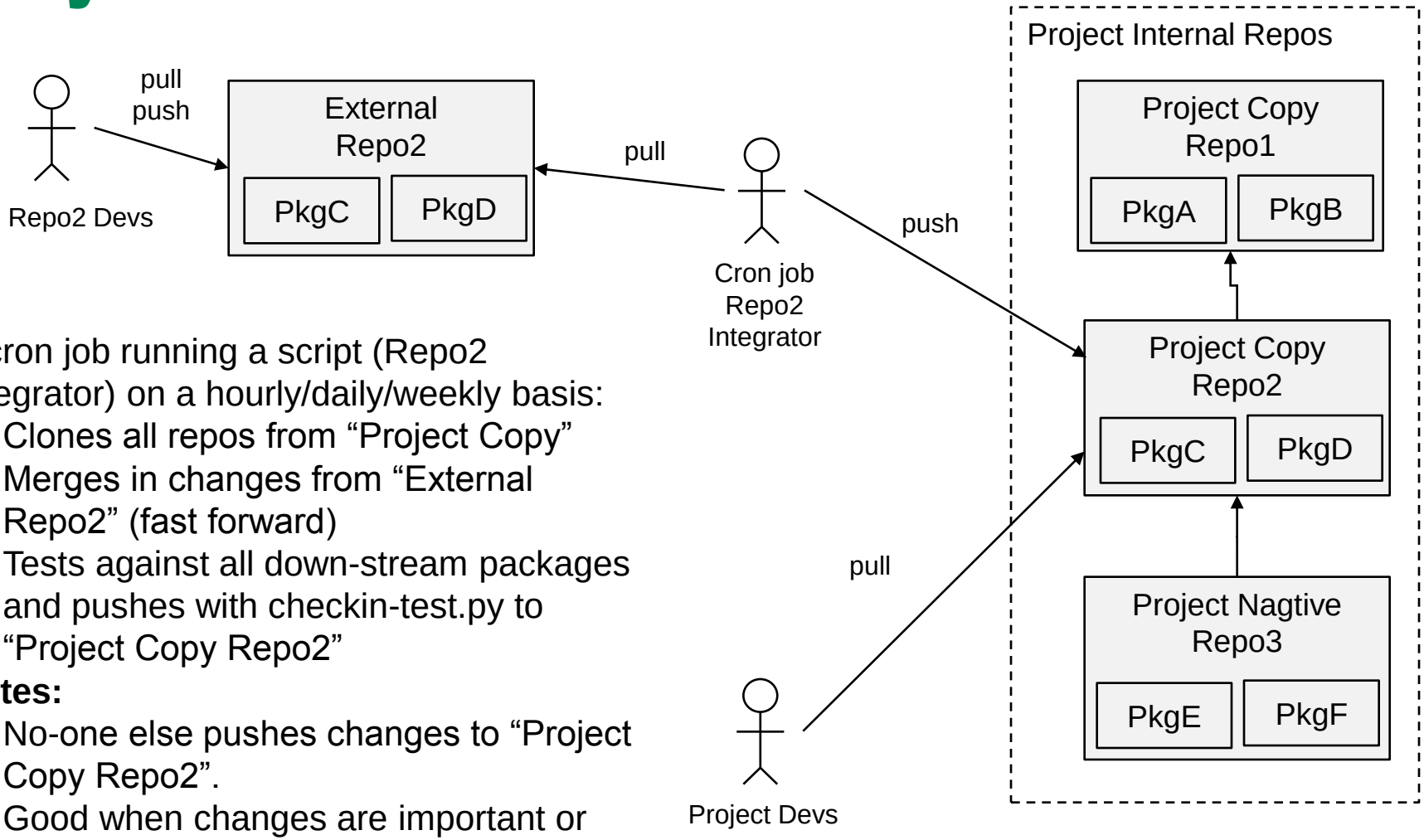
- Clones all repos from “Project Copy”
- Merges in changes from “External Repo2” (fast forward)
- Tests against all down-stream packages and pushes with checkin-test.py to “Project Copy Repo2”

Notes:

- No-one else pushes changes to “Project Copy Repo2”
- Good when changes are **not** urgent for Project or when “External Repo2” is unstable

VERA Repos: DataTransferKit, MAMBA, MOOSE

External repo is synced automatically using sync server



A cron job running a script (Repo2 Integrator) on a hourly/daily/weekly basis:

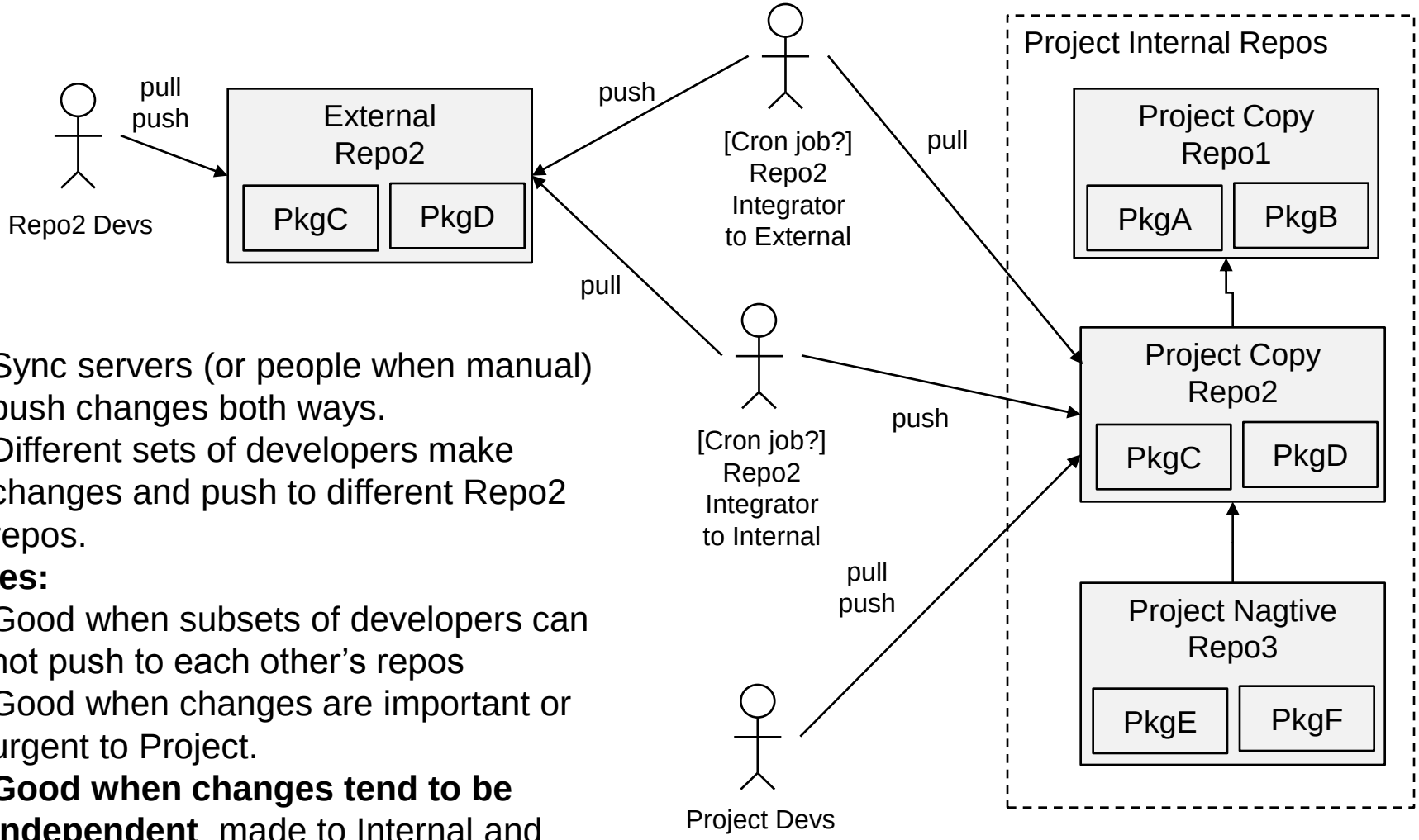
- Clones all repos from “Project Copy”
- Merges in changes from “External Repo2” (fast forward)
- Tests against all down-stream packages and pushes with checkin-test.py to “Project Copy Repo2”

Notes:

- No-one else pushes changes to “Project Copy Repo2”.
- Good when changes are important or urgent to Project and “External Repo2” is fairly stable.

VERA Repos: SCALE/Exnihilo, MPACT

Both external and internal repos pushed to



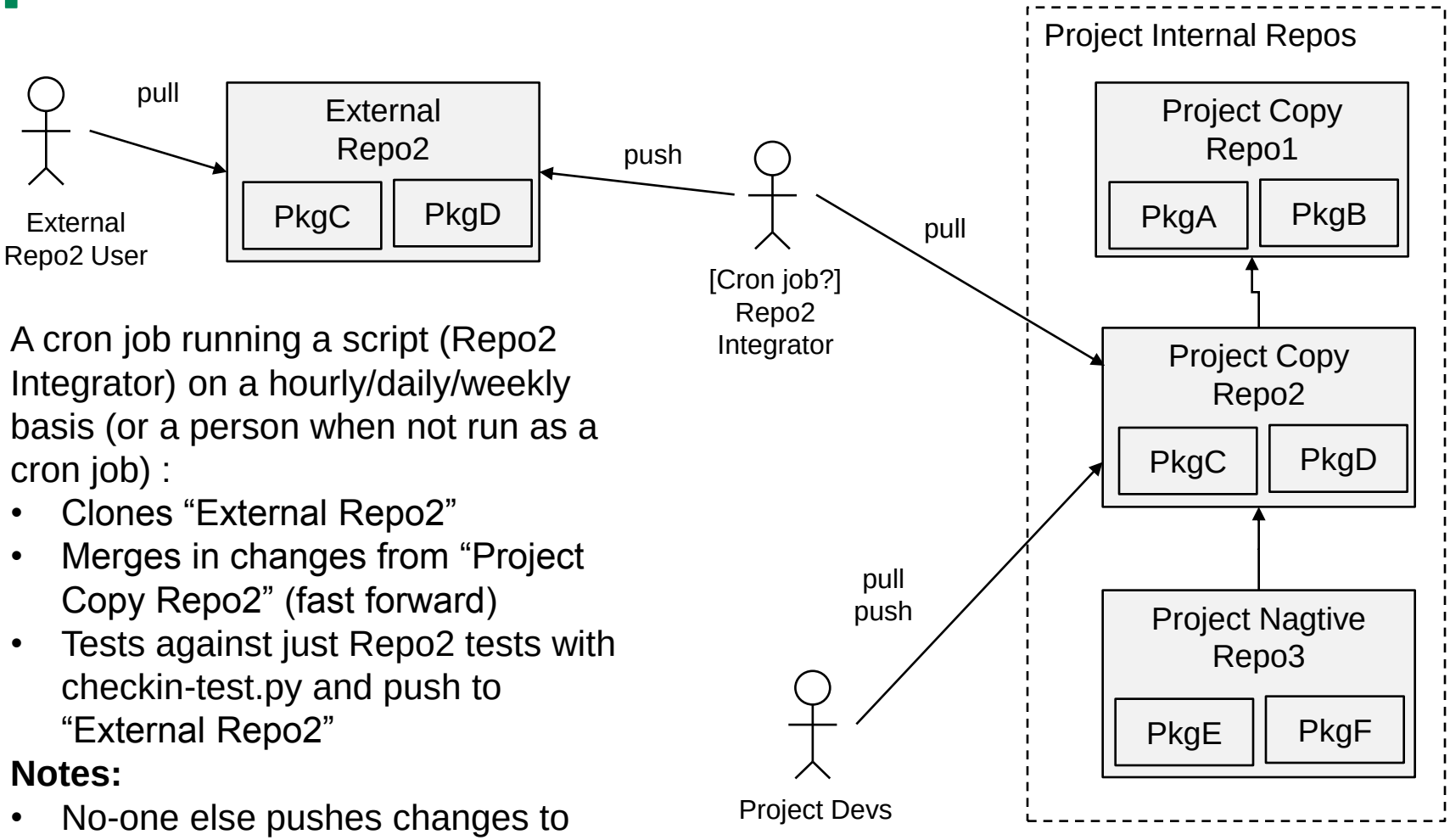
- Sync servers (or people when manual) push changes both ways.
- Different sets of developers make changes and push to different Repo2 repos.

Notes:

- Good when subsets of developers can not push to each other's repos
- Good when changes are important or urgent to Project.
- **Good when changes tend to be independent** made to Internal and External repos.
- Most complex and danger of merge conflicts that someone has to resolve!

VERA Repos: TriBITS, Trilinos

Internally managed repo is synced to an external repo



A cron job running a script (Repo2 Integrator) on a hourly/daily/weekly basis (or a person when not run as a cron job) :

- Clones “External Repo2”
- Merges in changes from “Project Copy Repo2” (fast forward)
- Tests against just Repo2 tests with checkin-test.py and push to “External Repo2”

Notes:

- No-one else pushes changes to “External Repo2”
- Good when you just want to make changes available to external users on a continuous basis.

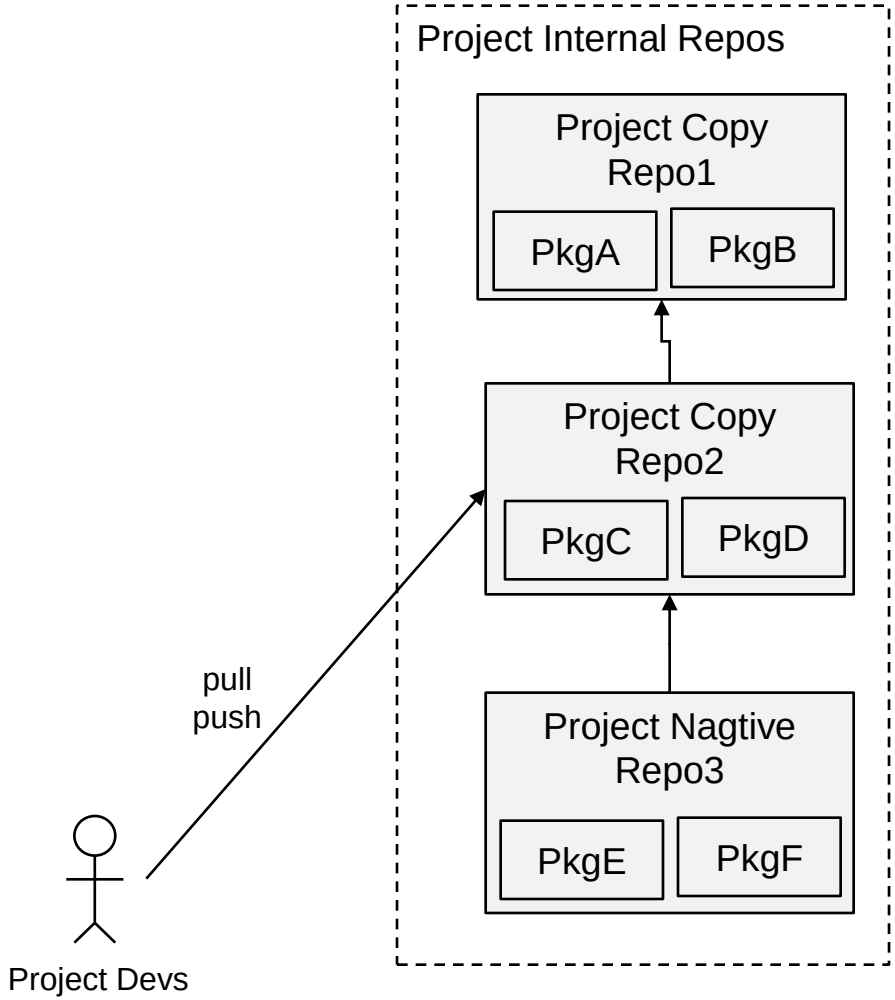
VERA Repos: COBRA-TF, TeuchosWrappersExt, VERAInExt

Internally managed repo

- Simple single-repo development model

Notes:

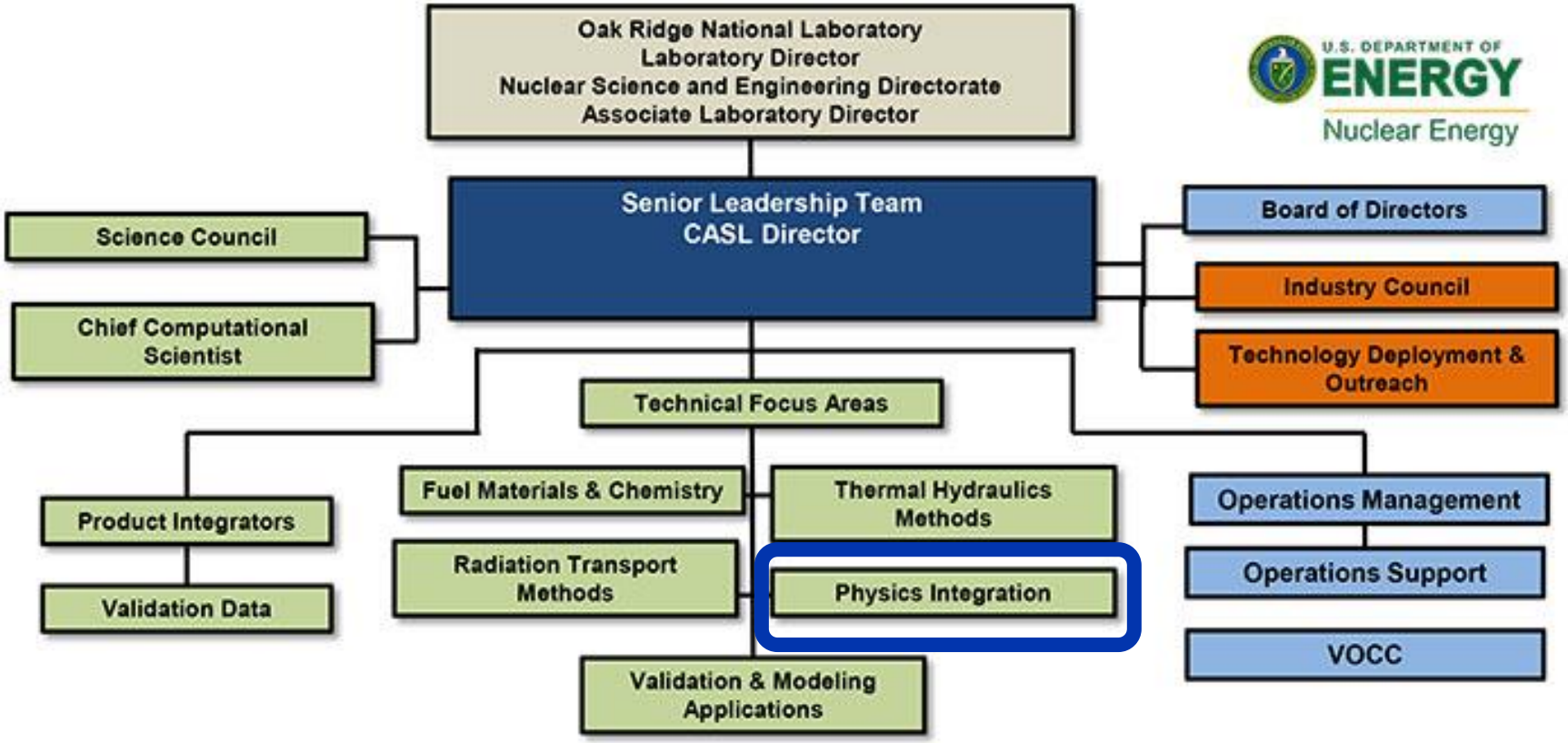
- Good when you don't need to coordinate with developers outside of Project



VERA Repos: MOOSEExt, PSSDriversExt , DakotaExt, VUQDemos

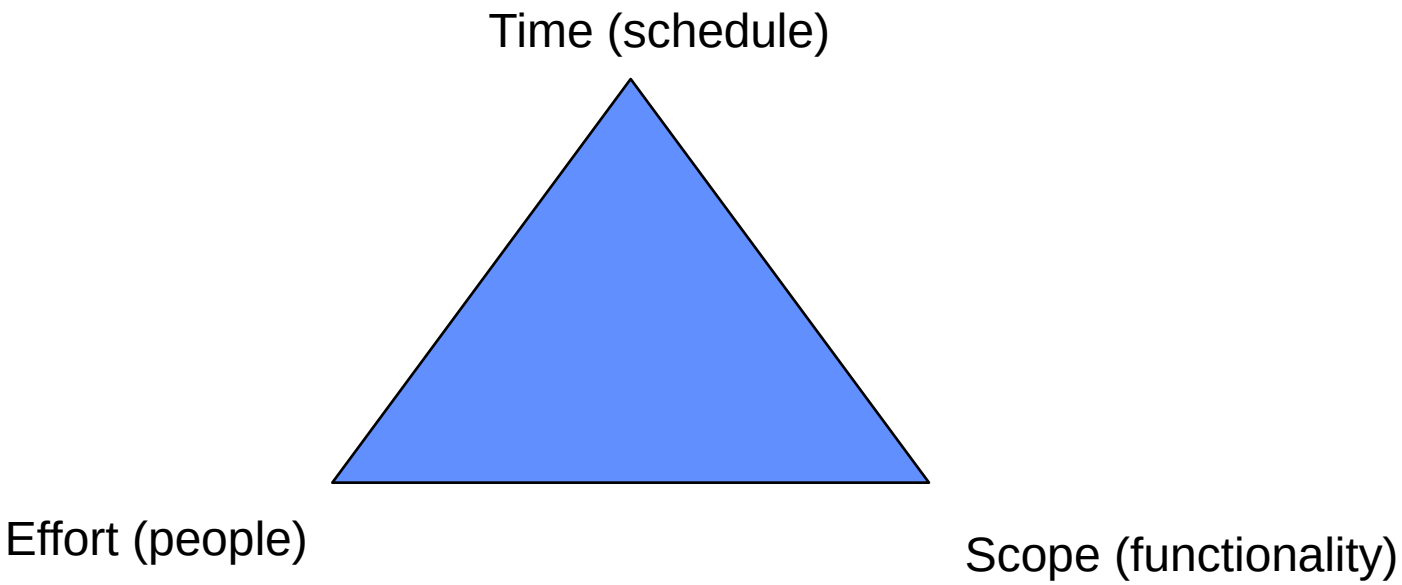
Lean/Agile Project Management

Organization of CASL



http://www.casl.gov/organization.shtml

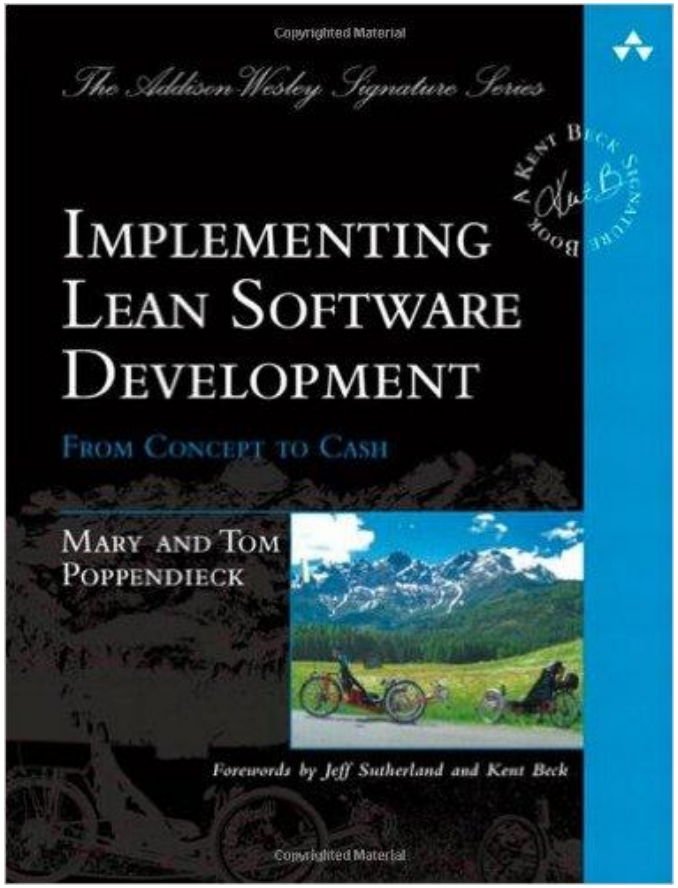
The Iron Triangle



- Given fixed Scope, Time is only weakly improved by adding more Effort!
  - => “Mythical Man Month”, Fred Brooks, 1975
- For any non-trivial software project it is impossible to fix all three (Time, Effort and Scope)
- Agile methods fix Time (fixed iterations, fixed releases) and Effort (fixed time size) and vary Scope (functionality) based on iterative feedback

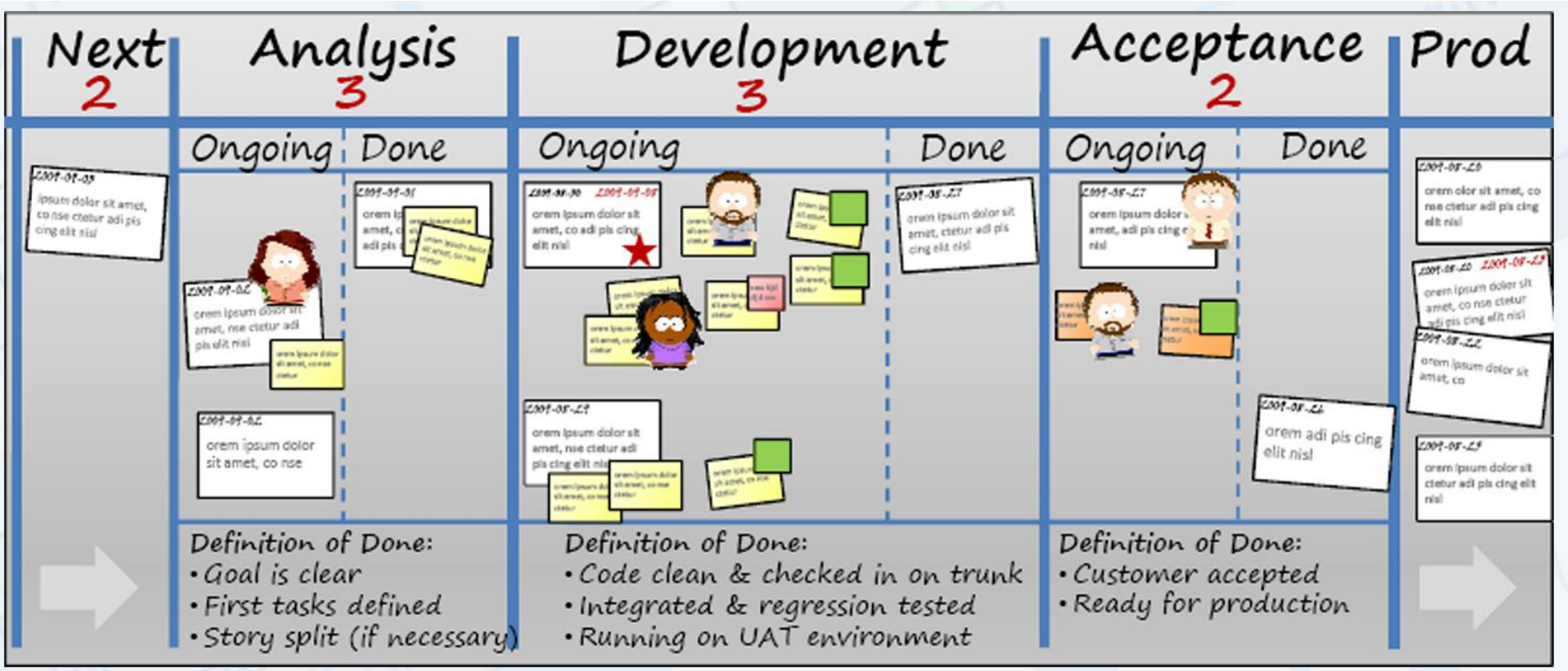
Overview of Lean Methods (7 Principles of Lean)

- Motivation for Lean Software Methods:
  - Lean Manufacturing (e.g. Toyota Production System)
- Seven Principles of Lean:
  1. Eliminate Waste
  2. Build Quality In
  3. Create Knowledge
  4. Defer Commitment
  5. Deliver Fast
  6. Respect People
  7. Optimize the Whole



Poppendieck, Mary and Tom. *Implementing Lean Software Development. Addison Wesley, 2007*

Overview of Kanban



- Kanban prescribes:
  - Just-in-time pull scheduling
  - Visualize workflow
  - Limit work in progress
  - Focusing on minimizing “cycle time”

CASL PHI Kanban Process

- **Requirements flow from CASL Milestone process** (6-month plans of record)
- **CASL milestones have some agile flexibility**:
  - L3 milestones can be moved back, re-scoped, even canceled based on an agile feedback.
  - DOE-reportable milestones can't be moved (since DOE is not agile), but scope can be adjusted based on agile feedback
- **PHysics Integration (PHI) focus area Kanban Process**:
  - **Customized Trac site** using Trac tickets for Epics, Stories & Tasks
  - **Projects and Product Owners**:
    - ***Infrastructure***: Build/test infrastructure, repository development workflow and integration strategies, deployment infrastructure
      - Product Owner: Roscoe Bartlett (ORNL)
    - ***Coupled Methods and Development***: Physics and mathematical coupling methods and related development
      - Product Owner: Roger Pawlowski (SNL)
    - ***VERA-CS***: Physics development, benchmark problem development and testing
      - Product Owner: Scott Palmtag (Independent)
  - **Weekly Standup Meetings**, one for each PHI Project
  - **Review/Retrospective/Planning (RRP) Meetings**:
    - Approximately every 2 to 3 months (flexible)

Summary

CASL VERA Development Challenges

- Large legacy codes under active development being constantly updated
  => Memory leaks, memory access errors, etc.
- Long configure and build times, unnecessary rebuilds after configure, etc.
- Non-native configure/build of MOOSE not using TriBITS CMake
- Stack of required TPLs:
  - MPI, LAPACK/BLAS, Boost, Zlib, PETSC, HDF5, QT, SILO
  - Different problems with different TPLs on different platforms
  - QT (required by SCALE) almost always hard to build and install
- Testing:
  - Insufficient unit and verification testing in legacy codes
  - Many full system acceptance tests that exist require 1000s of processors and hours to run. (currently not automated).
- Insufficient staff for infrastructure, testing, deployments, support etc.

Summary

CASL VERA:

- CASL almost continuously integrates development efforts from several independent teams and institutions to provide stream of stable versions of versions of VERA software.
- By controlling the build, integration, test and distribution Infrastructure one can:
  - Automate and enforce many SE best practices which helps to …
  - Avoid mistakes even with low developer training
  - Improve development productivity (by avoiding broken code)
  - Improve collaboration between groups (because changes are integrated almost continuously)

TriBITS:

- CASL VERA development has pushed multi-repo support in TriBITS
- TriBITS offers a lot of flexibility in assembling TriBITS Repositories and Packages into different TriBITS (complete CMake) projects.
- Major remaining issues yet to be resolved in TriBITS:
  - Combining concepts of packages and TPLs for large meta-projects
  - Finish support for wrapping externally configured/build software as TriBITS packages.
  - High-level and tutorial documentation
- But once these are done => TriBITS will be a good candidate for a universal meta-build and installation system for a **very** large amount of CSE software!

THE END

- **Contact:** bartlettra@ornl.gov
- **Sponsors:**
  - CASL: Consortium for the Advanced Simulation of Lightwater reactors