TriBITS

Tribal Build, Integrate, and Test System

Roscoe A. Bartlett (bartlettra@ornl.gov)
Computational Engineering and Energy Sciences Group,
Oak Ridge National Laboratory

OAK RIDGE National Laboratory

U.S. DEPARTMENT OF ENERGY

The Problem => Develop and Deploy Complex Software

- **Multiple software repositories and distributed development teams**
- **Multiple compiled programming languages (C, C++, Fortran) and mixed-language programs**
- **Multiple development and deployment platforms (Linux, Windows, Super-Computers, etc.)**
- **Stringent software quality requirements**

Solution Approach

=> TriBITS custom CMake build & test framework

Raw CMake vs. TriBITS

Raw CMakeLists.txt File

```
# Build and install library
SET(HEADERS hello_world_lib.hpp)
SET(SOURCES hello_world_lib.cpp)
ADD_LIBRARY(hello_world_lib ${SOURCES})
INSTALL(TARGETS hello_world_lib DESTINATION lib)
INSTALL(FILES ${HEADERS} DESTINATION include)

# Build and install user executable
ADD_EXECUTABLE(hello_world hello_world_main.cpp)
TARGET_LINK_LIBRARIES(hello_world hello_world_lib)
INSTALL(TARGETS hello_world DESTINATION bin)

# Test the executable
ADD_TEST(test ${CMAKE_CURRENT_BINARY_DIR}/hello_world)
SET_TESTS_PROPERTIES(test PROPERTIES PASS_REGULAR_EXPRESSION "Hello World")

# Build and run some unit tests
ADD_EXECUTABLE(unit_tests hello_world_unit_tests.cpp)
TARGET_LINK_LIBRARIES(unit_tests hello_world_lib)
ADD_TEST(unit_test ${CMAKE_CURRENT_BINARY_DIR}/unit_tests)
SET_TESTS_PROPERTIES(unit_test  PROPERTIES PASS_REGULAR_EXPRESSION "All unit tests passed")
```

TriBITS Package CMakeList.txt File

```
TRIBITS_PACKAGE(HelloWorld)
TRIBITS_ADD_LIBRARY(hello_world_lib
  HEADERS hello_world_lib.hpp SOURCES hello_world_lib.cpp)
TRIBITS_ADD_EXECUTABLE(hello_world NOEXEPREFIX SOURCES hello_world_main.cpp
  INSTALLABLE)
TRIBITS_ADD_TEST(hello_world NOEXEPREFIX PASS_REGULAR_EXPRESSION "Hello World")
TRIBITS_ADD_EXECUTABLE_AND_TEST(unit_tests SOURCES hello_world_unit_tests.cpp
  PASS_REGULAR_EXPRESSION "All unit tests passed")
TRIBITS_PACKAGE_POSTPROCESS()
```

- Library linking automatically handled
- Avoid duplication and boiler-plate code
- Fewer commands
- Install by default (most common)
- Automatic namespacing of test & exec names

CMake and TriBITS

Why CMake?

- Open-source tools maintained and used by a large community and supported by a profession software development company (Kitware).
- **CMake:**
  - Simplified build system, easier maintenance
  - Improved mechanism for extending capabilities (CMake language)
  - Support for all major C, C++, and Fortran compilers.
  - Automatic full dependency tracking (headers, src, mod, obj, libs, exec)
  - Faster configure times (e.g. > 10x faster than autotools)
  - Shared libraries on all platforms and compilers
  - Support for MS Windows (e.g. Visual Studio projects)
  - Portable support for cross-compiling
  - Good Fortran support (parallel builds with modules with src => mod => object tracking, C/Fortran interoperability, etc.)
- **CTest:**
  - Parallel running and scheduling of tests and test time-outs
  - Memory testing (Valgrind)
  - Line coverage testing (GCC LCOV)
  - Better integration between the test system and the build system

Why TriBITS?

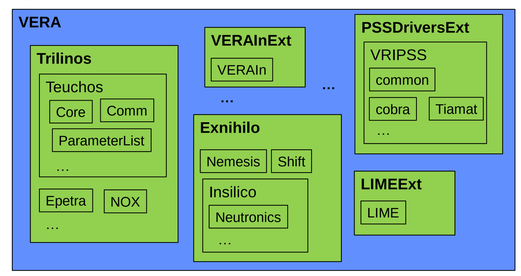
- Framework for large, distributed multi-repository CMake projects
- Reduce boiler-plate CMake code and enforce consistency across large distributed projects
- Subproject dependencies and namespacing architecture (packages)
- Automatic package dependency handling
- Additional tools for agile software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

TriBITS Structural Units

- TriBITS Project:
  - Complete CMake "Project"
  - Overall projects settings
- TriBITS Repository:
  - Collection of Packages and TPLs
  - Unit of distribution and integration
- TriBITS Package:
  - Collection of related software & Tests
  - Lists dependencies on SE Packages & TPLs
  - Unit of testing, namespacing, documentation, and reuse
- TriBITS Subpackage:
  - Partitioning of package software & tests
- TriBITS TPLs (Third Party Libraries):
  - Specification of external dependency (libs)
  - Required or optional dependency
  - Single definition across all packages

Packages + Subpackages = Software Engineering (SE) Packages

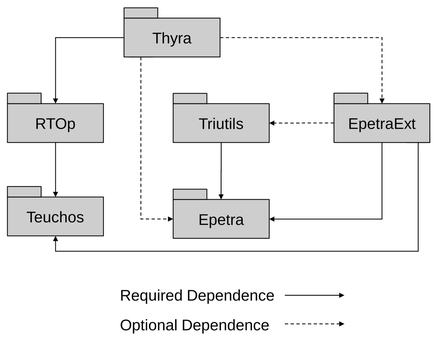
Example: VERA Meta-Project, Repositories, Packages & Subpackages



- **VERA:** Git repository and TriBITS meta-project (contains no packages)
- **Git repos and TriBITS repos**: Trilinos, VERAInExt, LIMEExt, Exnihilo, …
- **TriBITS packages**: Teuchos, Epetra, VERAIn, Insilico, LIME, VRIPSS, …
- **TriBITS subpackages**: TeuchosCore, InsilicoNeutronics, VRIPSSTiamat, …
- **TriBITS SE (Software Eng.) packages**: Teuchos, TeuchosCore, VERAIn, Insilico, InsilicNeutronics, …

Automated Package Dependency Handling

Package Dependency Structure (Example: Trilinos)



Package Dependencies.cmake Files

Teuchos

```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_TPLS  BLAS  LAPACK
  LIB_OPTIONAL_TPLS Boost  )
```

Epetra

```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_TPLS  BLAS  LAPACK  )
```

RTOp

```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_PACKAGES  Teuchos  )
```

Triutils

```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_PACKAGES  Epetra  )
```

EpetraExt

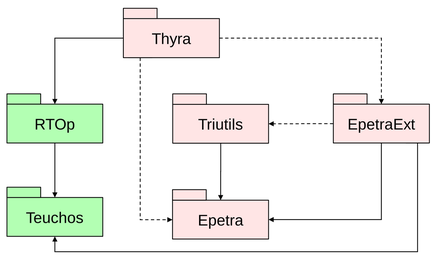
```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_PACKAGES  Epetra  Teuchos
  LIB_OPTIONAL_PACKAGES  Triutils  )
```

Thyra

```
TRIBITS_PACKAGE_DEFINE_DEPENDENCIES(
  LIB_REQUIRED_PACKAGES  RTOp  Teuchos
  LIB_OPTIONAL_PACKAGES  EpetraExt  Epera  )
```

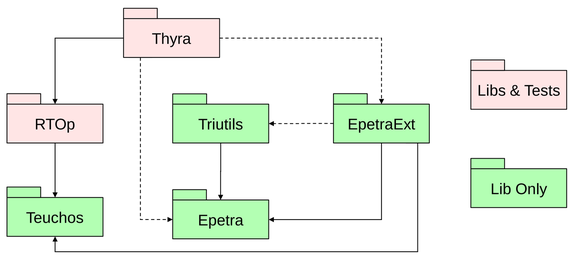
Pre-Push Testing: Change Epetra

```
$ ./do-configure \
  -D Trilinos_ENABLE_Epetra:BOOL=ON \
  -D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES:BOOL=ON \
  -D Trilinos_ENABLE_TESTS:BOOL=ON
```



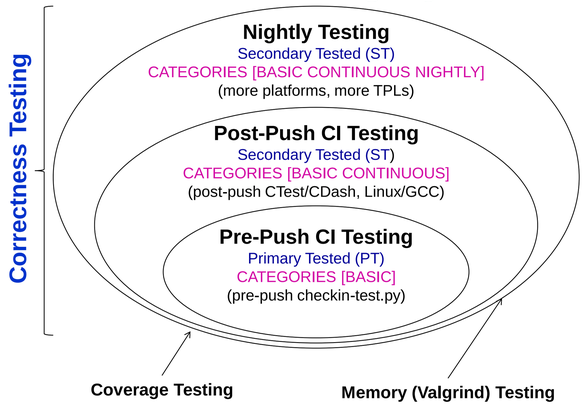
Pre-Push Testing: Change RTOp

```
$ ./do-configure \
  -D Trilinos_ENABLE_RTOp:BOOL=ON \
  -D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES:BOOL=ON \
  -D Trilinos_ENABLE_TESTS:BOOL=ON
```



Extended Testing Support

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()

CDash Dashboard for 4/6/2014

- Rolled-up summaries for each build case
- Nightly, CI, Experimental build cases

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!

TriBITS Miscellaneous Facts

- TriBITS System Dependencies:
  - TriBITS Core: Basic configure, build, test & install

    => Only raw CMake (2.8.4+)
  - TriBITS Extra SE tools (checkin-test.py, …)

    => Git (1.7.0.4+) and Python 2.4
- Usage of TriBITS:
  - Trilinos (SNL, originating project)
  - ORNL: SCALE, Exnihilo, DataTransferKit
  - Non-ORNL: MPACT (Univ. of Misc.), COBRA-TF (Penn. State)
  - CASL-Related: VERA
- TriBITS Development & Distribution:
  - 3-clause BSD-like license, Copyright SNL
  - Current: Trilinos (trilinos.sandia.gov), CASL (casl-dev)
  - Near future: **Github** (public repo, global pull)

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