TriBITS

Tribal Build, Integrate, and Test System

Roscoe A. Bartlett, Ph.D.

bartlettra@ornl.gov
http://web.ornl.gov/~8vt/
Computational Engineering and Energy Sciences Group,
Oak Ridge National Laboratory

SIAM Computational Science & Engineering Conference
Salt Lake City, Utah
March 14, 2015
The Challenge => 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
Overview of CASL VERA Development
Overview of CASL

• **CASL**: Consortium for the **Advanced Simulation** of **Lightwater** 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 **Approved by DOE and Congress!**
• 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)
CASL VERA Development Overview

• VERA Development is complicated in almost every way 😞
• VERA Currently Composed of:
  • 18 different git repositories on casl-dev.ornl.gov (clones of other repos) most with a different access list (NDAs, Export Control, IP, etc.)
• CMake build system using TriBITS 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 CI testing: 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

- Trilinos
  - TeuchosWrappersExt
  - VERAInExt
  - COBRA-TF
  - MPACT
  - PSSDriversExt
  - LIMEExt
  - DakotaExt
  - Dakota

- DatraTransferKit
  - SCALE
    - Exnihilo
    - MOOSE
    - MOOSEEExt

- hydrath
  - VUQDemos

Managed by UT-Battelle for the U.S. Department of Energy
Why CMake?

Why 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)
- Good Fortran support (parallel builds with modules with src => mod => object tracking, C/Fortran interoperability, etc.)
- Shared libraries on all platforms and compilers (support for RPATH)
- Faster configure times (e.g. > 10x faster than autotools)
- Native support for MS Windows (e.g. Visual Studio projects)
- Portable support for cross-compiling

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 namespaces 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
Raw CMake vs. TriBITS
Example Raw CMakeLists.txt File

```cmake
# 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(hello_world ${CMAKE_CURRENT_BINARY_DIR}/hello_world)
set_tests_properties(hello_world 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")
```

Executable and test names must be globally unique!
Example 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()

• Less duplication and boiler-plate code
• Fewer commands
• Build command wrappers:
  • Install by default (most common)
  • Optionally Install libraries and headers or just executables?
  • Optional global prefixing of libraries
  • And more …
• CTest command wrappers:
  • Automatic namespacing of tests and test executables
  • Classification of tests (BASIC, CONTINUOUS, NIGHTLY, …)
  • Uniform handling of timeouts (and scaling of timeouts)
  • And more …

Maintain consistency and add/change behavior across different independent repositories and packages and 1Ks of CMakeLists.txt files!
TriBITS Structural Units and Meta-Projects
### 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, naming, 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

\[\text{Packages + Subpackages} = \text{Software Engineering (SE) Packages}\]
Example: VERA Meta-Project, Repositories, Packages & Subpackages

- **VERA**: Git repository and TriBITS meta-project (contains no packages)
- **TriBITS and Git repos**: Trilinos, VERAInExt, COBRA-TF, MPACT, SCALE, Exnihilo ...
- **TriBITS packages**: Teuchos, Epetra, VERAIn, Insilico, COBRA_TF, MPACT_Drivers, ...
- **TriBITS subpackages**: TeuchosCore, InsilicoNeutronics ...
- **TriBITS SE packages**: Teuchos, TeuchosCore, VERAIn, Insilico, InsilicNeutronics, ...
The same TriBITS repositories can be arranged into multiple TriBITS projects.
Automated Package Dependency Handling
Package Dependency Structure (Example: Trilinos)
Package Dependencies.cmake Files

**Teuchos**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_TPLS BLAS LAPACK
  LIB_OPTIONAL_TPLS Boost )
```

**Epetra**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_TPLS BLAS LAPACK )
```

**RTOp**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_PACKAGES Teuchos )
```

**Triutils**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_PACKAGES Epetra )
```

**EpetraExt**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_PACKAGES Epetra Teuchos
  LIB_OPTIONAL_PACKAGES Triutils )
```

**Thyra**
```cmake
tribits_package_define_dependencies(
  LIB_REQUIRED_PACKAGES RTOp Teuchos
  LIB_OPTIONAL_PACKAGES EpetraExt Epera )
```
CI Testing: Change Epetra

$ ./do-configure \
-D Trilinos_ENABLE_Epetra=ON \n-D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES=ON \n-D Trilinos_ENABLE_TESTS=ON
CI Testing: Change RTOP

$ ./do-configure \
-D Trilinos_ENABLE_RTOP=ON \
-D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES=ON \
-D Trilinos_ENABLE_TESTS=ON
Multi-Repository Support
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?
tribits_project_define_extra_repositories(
    TriBITS         ""   GIT  git@casl-dev:TriBITS    ""  Continuous
    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
    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  ""  Continuous
    hydrath        ""   GIT  git@casl-dev:hydrath  ""  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, SCALE, MOOSE
$ ./clone_extra_repos.py

...
$ gitdist-status

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

(tip: to see a legend, pass in --dist-legend.)
Testing Support
TriBITS Standard Testing Layers

Nightly Testing
Secondary Tested (ST)
CATEGORIES [BASIC CONTINUOUS NIGHTLY]
(more platforms, more TPLs)

Post-Push CI Testing
Secondary Tested (ST)
CATEGORIES [BASIC CONTINUOUS]
(post-push CTest/CDash, Linux/GCC)

Pre-Push CI Testing
Primary Tested (PT)
CATEGORIES [BASIC]
(pre-push checkin-test.py)

Correctness Testing

Coverage Testing

Memory (Valgrind) Testing
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()

**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!
TriBITS Miscellaneous Facts and Future Work

• TriBITS System Partitioning and Dependencies:
  • TriBITS Core: Basic configure, build, test, install, and creating distributions
    => Only requires raw CMake 2.8.11+
    => 10K lines of CMake code (1M of disk space)
  • TriBITS CI Support (checkin-test.py, clone_extra_repos.py,…)
    => Requires Git (1.7.0.4+) and Python 2.4
  • See TriBITS Developers Guide for more details (http://tribits.org )

• 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
  • Main source hosted on GitHub (https://github.com/TriBITSPub/TriBITS )
  • Documentation hosted on http://tribits.org

• Near-term Future Work:
  • More flexibility on pre-building packages and linking in as TPLs
  • Define a standard installation of TriBITS
  • Put out a TriBITS release
  • Finish overview document and tutorials
  • More error checking to catch user mistakes
THE END

• **Contact:** bartlettra@ornl.gov

• **Sponsors:**
  - CASL: Consortium for the Advanced Simulation of Lightwater reactors