



A Practical Application of the Computational Science Environment (CSE)

by John Vines, Kelly Kirk, Eric Mark, Carrie Spear, and Joel Martin

ARL-TR-5840

December 2011

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005

ARL-TR-5840

December 2011

A Practical Application of the Computational Science Environment (CSE)

John Vines, Kelly Kirk, Eric Mark, Carrie Spear, and Joel Martin
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) December 2011		2. REPORT TYPE Interim		3. DATES COVERED (From - To) October 2010 to October 2011	
4. TITLE AND SUBTITLE A Practical Application of the Computational Science Environment (CSE)				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) John Vines, Kelly Kirk, Eric Mark, Carrie Spear, and Joel Martin				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-CIH-C Aberdeen Proving Ground, MD 21005				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-5840	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The Computational Science Environment (CSE) is a collection of open source software tools and utilities that encompass a large number of state-of-the-art application program interfaces (APIs) (i.e., Qt, Python, and SciPy). The CSE software development system fosters the development of modern software applications and is structured to support individuals, small teams, or large distributed development groups. An integral piece of CSE is its intrinsic support for software testing, which is required to verify and validate the functionality and results of all production software. CSE provides extensive software testing suites and quality assurance dashboards to post results. Extensibility is another core capability of the CSE; CSE has a dynamic environment that can be leveraged through "add-ons" to incorporate established applications and previously developed utilities. A good example of a CSE add-on has been developed for the High Performance Computing Modernization Program's (HPCMP) Multiscale Reactive Modeling (MSRM) Institute for the Multiple Object Evolutionary Strategies (MOES) code. The MSRM's Infrastructure team has worked closely with the CSE team and MOES developers to design a cross-platform build and testing system for the MOES code. The CSE MOES add-on provides the MSRM institute with the ability to use, develop, build, and test the entire MOES system.					
15. SUBJECT TERMS Computational science, computer science					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 18	19a. NAME OF RESPONSIBLE PERSON John Vines
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) (410) 278-9150

Contents

List of Figures	iv
Acknowledgments	v
1. Introduction	1
2. Configuration Management	1
2.1 The Software Repository	1
2.2 Software Configuration Management	2
3. CMake and Modules	2
4. Automated Build and Test	4
5. Data Analysis and Visualization	6
6. Conclusion	7
List of Symbols, Abbreviations, and Acronyms	8
Distribution List	9

List of Figures

Figure 1. The quality assurance dashboard reports the results of the test.....	5
Figure 2. Example plot of MOES data.....	6
Figure 3. Example plot of MOES data, zoomed to show more curve details.....	6

Acknowledgments

We would like to thank the High Performance Computing Modernization Program for supporting this work through computing resources.

INTENTIONALLY LEFT BLANK.

1. Introduction

The Computational Science Environment (CSE) provides users with a standard development environment as well as open source software tools and utilities for data analysis and visualization. A significant advantage of using CSE is its ability to provide researchers and engineers with a development foundation capable of supporting their software project. This is supported through a CSE subproject or “add-on.” CSE provides a straightforward framework for creating and automatically testing add-ons along with the ability to customize the build process by specifying particular compilers, optimization flags, and external libraries to meet the researcher’s needs. CSE add-ons are a simple way for researchers and developers to extend the common baseline set of tools, libraries, and applications in CSE. The CSE team has worked closely with the Multiple Object Evolutionary Strategies (MOES) developers as well as the Multiscale Reactive Modeling’s (MSRM) Infrastructure team to design a modular build and test system that simplifies the entire build process and make the projects easier to maintain and modify.

2. Configuration Management

2.1 The Software Repository

Source code management for the MSRM project is done through Git. Git is a distributed version control system unlike Subversion (SVN), which is a centralized version control system. Because Git is a distributed version control system, when a user performs a checkout (clone), every revision of every file is available as part of the local copy. An additional benefit of Git is branching. While branching is not unique to Git, one benefit is being able to completely remove a branch (there is no trace like there is in SVN), so branches can be easily created and discarded as necessary.

Git Repository permissions are handled at the operating system (OS) and filesystem level through the use of Linux groups and Filesystem action control lists (ACLs)*. Each project is designated a unique group name for the facilitation of permissions, only individuals designated by the project point of contact (POC) will be added to the group.

*The ACL is a list of permissions attached to an object. This list specifies which users or groups are granted permissions to an object and what actions can be performed on that object.

The following commands are used to provide a specific group with read and write permission to their repository:

```
setfacl -R -m g:cse:rwX cse_stable  
find repo -type d | xargs setfacl -R -m d:g:cse:rwX
```

There are two software repositories for each Atomistic add-on. Each add-on has a development repository and a production repository. All developers have read and write permissions to the development repository. Only individuals deploying the add-on have full access to the production repository. Each of these repositories is built, tested, and reported on nightly. The production repository is built, tested, and deployed on various high performance computing (HPC) systems as directed.

2.2 Software Configuration Management

Because the team is “geographically distributed,” an online interface was necessary to simplify communication. The team uses Redmine[†], an open-source, database-driven project management Web application. The Redmine application offers a substantial preconfigured solution that is easy to setup and can be customized to better fit a particular project. Redmine was chosen as it provides a number of features that are of interest for team interaction. The most significant features were the ability to support multiple projects, a customizable issue tracking system, e-mail and news notifications, Wiki support, access control, and integration with several source code management systems (e.g., Git, as described previously). Redmine’s tracking system provides a simple solution for managing bugs, general support issues, and requests for new features allowing the entire team to determine priorities, provide updates, and know the status of any outstanding items. As items in the tracking system are updated, the responsible personnel and others with an interest are informed via e-mail. Redmine also provides a common place for current source code via the connected Git repository, as well as code and user documentation via its Wiki and file storage features. Collectively, these features reduce the time needed to oversee and manage multiple projects, allow the team to keep track of issues specific to their assigned projects, and provide a common place for project related communication.

3. CMake and Modules

The CSE MOES add-on uses CMake to drive build process. CMake is an open-source cross platform build system with strong system introspection capabilities. There are numerous macros distributed with CMake to discover what packages are installed on a system, their version numbers, and where these packages are installed. Because the MOES code was needed by the team’s scientists on several different HPC architectures, the system introspection capabilities of

[†]Redmine Web site. <http://www.redmine.org> (accessed 2011).

CMake significantly simplified deployment. There is a top level CMakeLists.txt file that includes all of the sub-packages as well as the module installation code.

The following is from the top level CMakeLists.txt project it sets up the default installation prefix and includes all of the sub-package directories. The CSE provides template files for the top level CMakeLists.txt file:

```
# Add-on CSE_ADDON_CVS Packages
cmake_minimum_required(VERSION 2.8)
# The name of the build project
project(CSE_ADDON_MOES)
set(SUBPROJECT "moes")
set(HOME_DIR MOES)
include(CTest)

# Enable testing for the project
ENABLE_TESTING()

include(ExternalProject)
set(base "${CMAKE_BINARY_DIR}/CMakeExternals")
set_property(DIRECTORY PROPERTY EP_BASE ${base})

# Find the CSE installation on this system

find_path(CSE_HOME Release $ENV{CSE_HOME} /usr/cta/CSE $ENV{HOME}/CSE)
list(APPEND CMAKE_MODULE_PATH "${CSE_HOME}/Misc/CMake")
find_package(CSE)
# Default installation prefix is /home/userid, this can be changed at configure time
# To change the Install prefix cmake -DMOES_INSTALL_PREFIX=/prefix/you/want
set(MOES_INSTALL_PREFIX "$ENV{HOME}/${HOME_DIR}" CACHE PATH "Install Path")
set(CMAKE_INSTALL_PREFIX ${MOES_INSTALL_PREFIX} CACHE INTERNAL "" FORCE)

find_program(PATCH_PROGRAM patch
    PATHS /usr/bin)

# include all subdirectories
add_subdirectory(airebo)
add_subdirectory(lp_solve)
add_subdirectory(reac)
add_subdirectory(moes)
```

Modules are automatically built and installed with each CSE add-on. Modules are a command-line tool providing dynamic modification of a user's environment, thus eliminating the need for the user to modify their own environment variables (PATH, LD_LIBRARY_PATH, PYTHONPATH) to compile, test, and run code. The CSE team provides a default file for each module that is populated by package specific variable names at configure time.

The MOES project installs the following modules for use by MOES developers and individuals performing batch runs using the MOES code:

```
----- /usr/cta/CSE.atomistic.2011-04-22/MOES/modules -----  
  
cse-msrm-atomistic/airebo/1.1    cse-msrm-atomistic/moes/1.0  
cse-msrm-atomistic/airebo/latest cse-msrm-atomistic/moes/latest  
cse-msrm-atomistic/lp_solve/5.5  cse-msrm-atomistic/reac/1.2  
cse-msrm-atomistic/lp_solve/latest cse-msrm-atomistic/reac/latest
```

4. Automated Build and Test

CTest is a testing tool distributed with CMake. This tool can be used to automate building and testing of a project. CTest can also submit testing results to a dashboard for display and review. CDash is a product distributed with Kitware's CMake. CDash is a Web-based, open-source software testing server. CDash organizes and displays testing results on a simple, easy to understand Web page. The immediate feedback that developers receive on the dashboard helps to encourage careful testing and code review prior to submitting code modifications to the repository.

In order to submit to a local dashboard it is necessary to have a CTestConfig.cmake file. The following is a basic sample of a CTestConfig.cmake file:

```
set(CTEST_PROJECT_NAME "MOES_Development")  
set(CTEST_NIGHTLY_START_TIME "21:00:00 EDT")  
#use https  
set(CTEST_DROP_METHOD "https")  
set(CTEST_DROP_SITE "your.web.dash.site")  
set(CTEST_DROP_LOCATION "/CDash/submit.php?project=MOES_Development")  
set(CTEST_DROP_SITE_CDASH TRUE)  
set(CTEST_CURL_OPTIONS  
"CURLOPT_SSL_VERIFYPEER_OFF;CURLOPT_SSL_VERIFYHOST_OFF")
```

For the CSE MOES add-on, each package is tested individually and the entire project is also tested as a whole. Because all of the packages in MOES use CMake to build, all of the regression tests leverage CTest. The CTest output is stored locally in an extensible markup language (XML) file that gets submitted to the CDash quality assurance dashboard upon completion of the builds and tests. In order to ensure that software modifications did not impact numerical accuracy, test results are automatically validated against a known set of values. This verification is done through a bash script that gets executed by CTest; value tolerances can be set in this script if necessary. The following line needs to be added to the CMakeLists.txt file enable testing for the project:

```
enable_testing()
```

The CMake tests are added to the build simply by adding the following lines to the CMakeLists.txt file, these lines call the bash scripts that execute the tests:

```
add_test(TestMoesCalculate test_moes.sh)
```

```
add_test(TestAireboDriver test_airebo.sh)
```

Figure 1 shows the quality assurance dashboard reports the results of the test.

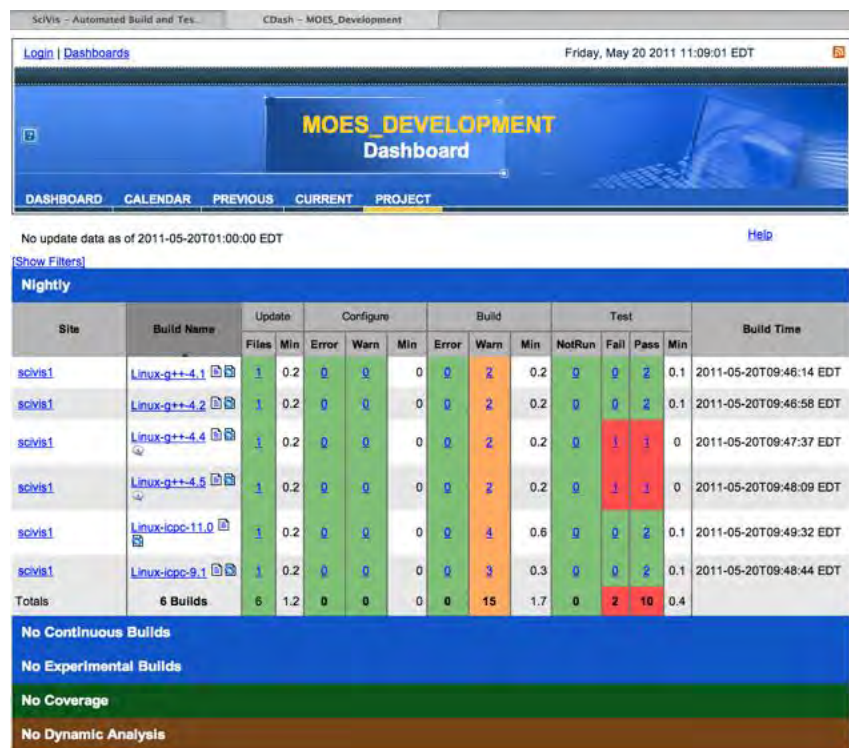


Figure 1. The quality assurance dashboard reports the results of the test.

5. Data Analysis and Visualization

Using several of the tools available in CSE (Python, NumPy, PyQt, and Matplotlib), the CSE team worked with the MSRM Infrastructure team and the MOES developers to assist with the creation of data conversion and visualization programs. A Python program was developed to translate the MOES output code into a CSV file that can be used by numerous spreadsheet and graphing packages. A Python program was also developed to automatically generate graphs of the data output using Python, NumPy, and Matplotlib.

Figures 2 and 3 shows example plots of the MOES data.

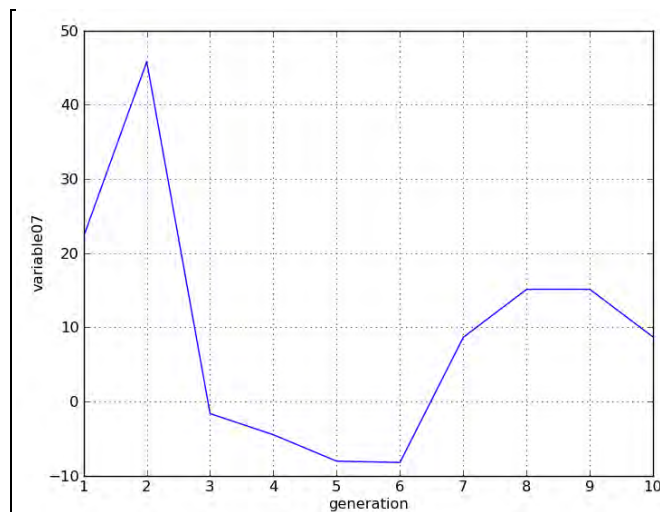


Figure 2. Example plot of MOES data.

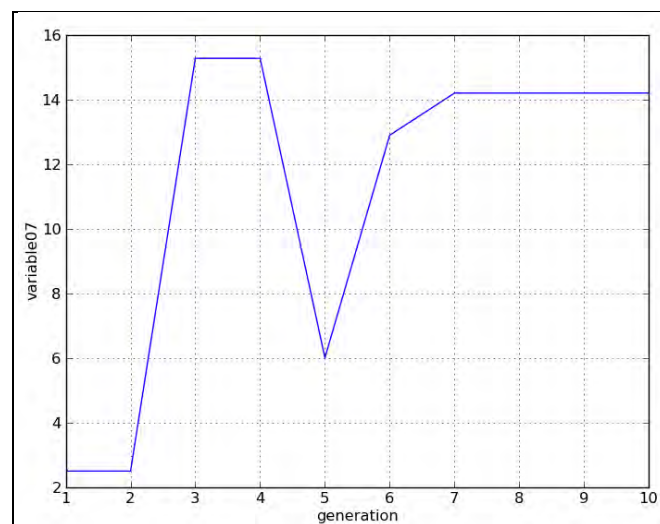


Figure 3. Example plot of MOES data, zoomed to show more curve details.

6. Conclusion

CSE offers a comprehensive software environment that is flexible, provides modern software application program interfaces (APIs), and supports development efforts scaling from individuals to large geographically dispersed teams. An integral part of the CSE is the automated building, testing, and reporting system for all software development projects. The CSE “add-on” framework allows users and developers to create domain-specific capabilities outside of the system-level architecture.

List of Symbols, Abbreviations, and Acronyms

ACLs	action control lists
APIs	application program interfaces
CSE	Computational Science
HPC	high performance computing
HPCMP	High Performance Computing Modernization Program
MOES	Multiple Object Evolutionary Strategies
MSRM	Multiscale Reactive Modeling
OS	operating system
POC	point of contact
SVN	Subversion
XML	extensible markup language

- 1 DEFENSE TECHNICAL
(PDF INFORMATION CTR
only) DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218
- 3 US ARMY RSRCH LAB
ATTN IMNE ALC HRR
MAIL & RECORDS MGMT
ATTN RDRL CIO LL TECHL LIB
ATTN RDRL CIO MT TECHL PUB
ADELPHI MD 20783-1197
- 9 US ARMY RSRCH LAB
ATTN RDRL CIH
D THOMPSON
ATTN RDRL CIH C
J VINES
E MARK
K KIRK
J CLARKE
ATTN RDRL CIH M
M KNOWLES
C SPEAR
J MARTIN
ATTN RDRL CIH S
L BRAINARD

INTENTIONALLY LEFT BLANK.