SANDIA REPORT

SAND2018-6880
Unlimited Release
Printed June 26, 2018

Sierra Structural Dynamics Code Verification Plan

Sierra Structural Dynamics Development Team

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

SAND2018-6880
Unlimited Release
Printed June 26, 2018

Sierra Structural Dynamics Code Verification Plan

Sierra/SD Development Team

Abstract

Verification and validation (V&V) of scientific computing programs are important at Sandia National Laboratories (SNL) due to the expanding role of computational simulation in managing the United States nuclear stockpile. This document presents the verification plan for the Sierra Structural Dynamics application. This verification plan includes code development practices, structure of test suites, and additional quality assurance practices used by the Sierra/SD team.

Contents

Chapter 1

Sierra/SD Verification Procedures

1.1 Overview

This document contains a verification overview for the software package Sierra/SD. In contrast to the Sierra/SD User's Manual,[?] which demonstrates how to use the code, and the Theory Manual,[?] which details the underlying mathematics of the code, the verification manual is a list of well documented verified examples demonstrating how the code performs on a subset of verification problems. In additional to the verification tests detailed in this document high confidence in the correctness of Sierra/SD is maintained by an extensive test suite, several code quality tools, and rigorous team processes. The intent is to fully verify each capability in Sierra/SD. This manual should be used to gain a level of confidence in the rigor for which Sierra/SD is verified for high consequence analysis. However, quality verification is a journey of continuous improvement. There may be gaps in the verification coverage. If there is a clear gap in the verification coverage that is essential to analysis, the Sierra/SD team should be contacted at sierra-help@sandia.gov.

1.2 Code Development Practices

The first step to a well verified code is code development practices that ensure all new code features are properly tested. The Sierra/SD team follows the laws of test driven development (TDD) coding practice as outlined in Clean Code.[?] The three laws of TDD are

1. You may not write production code until a failing unit test is written.
2. You may not write more of a unit test than is sufficient to fail.
3. You may not write more production code than is sufficient to fix the currently failing test.

Following these laws ensures that all new capability is covered by tests, and that all capability modified through user stories or corrected by user support is also covered by tests. However, these practices fail to ensure that *all* legacy capability is adequately covered, or that all permutations of capability are well verified. The Sierra/SD process for covering permutations of capability is outlined in 1.8. In addition to the enumerated TDD practices the Sierra/SD development team also uses code reviews, pair programming, and external beta testing as additional safeguards to prevent coding errors.

1.3 Overview of testing Pyramid

In order to efficiently maintain code quality a properly organized suite of tests must be used, a large number of small tests of individual capabilities building up to smaller numbers of large and complex tests. There are many types of tests for Sierra/SD: Unit, Fast (Continuous), Performance, Verification, Regression, and Acceptance. For tests to have value they most be run regularly and in an automated fashion. With the exception of a few large acceptance tests the entire Sierra/SD test suite is run nightly.

- *Unit Tests:* a test of an individual source code function. Unit tests are generally run through the Google GTEST framework. A unit test can be used to verify a given function has the correct behavior for every possible input. Unit tests are very fast. Sierra/SD currently uses many thousand unit tests.

- *Fast Tests:* a test that must run in under ten seconds. Fast tests are run every hour on the master branch of the Sierra code base. This high run frequency allows quickly pinpointing any issues introduced into the code base. The fast test suite is designed to give a broad coverage of all core Sierra/SD features. Sierra/SD uses about a thousand fast tests.

- *Verification Tests:* a test that compares test outputs to an analytic result or confirms the test has some expected property (such as a convergence rate.) Verification tests are one of the most valuable test types and the verification test suite will continue to be expanded over time. Sierra/SD maintains about a thousand verification tests.

- *Regression Tests:* a test that confirms the code produces and expected output, but without rigorous mathematical demonstration that the output is indeed correct. Generally a test case is produced and then engineering judgment used to confirm the test case is behaving as expected. The test then confirms this approved behavior is maintained. An example would be the modal decomposition of a complex shape part. Currently Sierra/SD uses several thousand regression test. Regression tests are a necessity, but the the Sierra/SD development team is moving over time to a larger balance of tests in the more valuable unit and verification categories.

- *Performance Tests:* a test used to confirm Sierra/SD maintains acceptable runtime and memory use bounds. These tests are expensive and Sierra/SD maintains only about a hundred.

- *Acceptance Tests:* a test of a full analysis use case provided by an analyst. Acceptance tests are the largest and most complex tests in the system. An acceptance test ensures the work flow for an entire complex analysis chain maintains functionality. As acceptance tests are very expensive Sierra/SD maintains only about a dozen to cover the most important and commonly used work flows.

1.4 User Support Process

The key to credible capability is a user support process that identifies, patches, and tests against any bugs found by Sierra/SD analysts. When a bug report is submitted a minimal representative example of the bug is produced by the developers and added as a test to the nightly test suite. After necessary development is done to resolve the issue the new nightly test ensures that the bug will not reappear in future releases.

1.5 Verification Policy for New Features

When new capability is added to Sierra/SD, the code development processes outlined in Clean Code[?] and Test Driven Development are followed. The new development *always* begins with a unit testing of new functionality. After completing the unit test, a self-documenting verification test is added that demonstrates the capability reproduces an analytical result. Additionally, regression tests may added that exercise the range of inputs of the capability. Once these tests are in place, an acceptance model, received from key analyst stakeholders, is run to ensure the capability behaves as expected and gives an acceptable result.

The Sierra/SD team migrated to a structure of individual test documentation maintained in the test repositories in 2013. The legacy formats are also included in this document, and eventually will be migrated to the new format. Thus though all verification tests are verified to a high level of rigor, not all verification tests are included in this verification test manual.

1.6 Nightly Testing Process

Every night the entire code base is compiled on multiple platforms with multiple compilers. Some subset of the nightly tests are run on each platform. Every fast and nightly test is run on the development platform, compiled with both debug/release and gcc/intel compilers. Additionally, all nightly tests are run on the Trinity surrogate (both Haswell and Knights Landing chips). The entire test suite (including performance tests) are run on intel-release on the primary HPC production platform dedicated to Nuclear Deterrence. Some subsection of the tests are run on experimental platforms, such as Darwin (MAC-OS), Broadwell, and Ride (GPU). These tests are useful because they may identify software quality issues that don't cause problems in the production platforms, but could in the future as new platforms move into production.

1.7 Other SQA Tools

In addition to the nightly testing process, other software quality tools are run nightly to check for possible code errors or gaps in testing coverage. These tools include the memory checker Valgrind, the Feature Coverage Tool (FCT), and the Line Coverage Tool (LCOV).

1.7.1 Valgrind

Valgrind is a tool used to check for memory leaks and memory errors. A memory leak is when memory is allocated, but never freed while the program is still running. The existence of memory leaks within loops can lead to a simulation taking an increasing amount of memory as simulation time increases, eventually leading to code failure. A memory error represents the executable accessing memory that has not been allocated, or is otherwise out of bounds. A memory error generally results in unpredictable behavior, and can lead to fatal segmentation faults. Valgrind is run nightly on both the "nightly" and "fast" tests. All memory leaks and errors are eliminated for every sprint snapshot and release version of Sierra/SD.

1.7.2 LCOV

The coverage tool For Sierra/SD, LCOV, measures the code source line coverage of unit, fast, and nightly testing. The LCOV tool reports how many times each line of code is called for the respective test suite. For each file, folder, and executable in Sierra LCOV reports the percentage of lines in the code that are covered by at least one test. For example, as of the 4.48 release, unit tests cover 48.3%, fast tests cover 79.4%, and nightly tests cover 86.0% of the code base. It is up to the development team to ensure that all new features are well covered. The Sierra/SD development team strives to improve test code coverage over time. However, 100.0% coverage is not always practical. Some uncovered code is either non-released research capability or depreciated legacy capability. Additionally many error messages do not have a test that hits the error message, therefore the line of code with the error message may be uncovered.

1.8 FCT

For Sierra/SD the Feature Coverage Tool (FCT) creates three documents from an input file; the annotated input file, the two way coverage graph and the list of best matching tests.

The FCT can be used by analysts to assess the Sierra/SD verification rigor for a specific analysis. Additionally the Sierra/SD development team can use output of the FCT prioritize needs for verification test suite improvement.

The annotated input file shows the features (corresponding to input deck lines) that are used in verification tests (in green), regression tested (yellow) or untested (red). Developers and analysts can use this tool to see if for an analysis in question untested features are used and take action to mitigate or explain them. One mitigation strategy is to create a new verification test for the feature. An explanation is needed if the FCT has indicated a false positive (the FCT tool is helpful, but still in development).

The second document produced by FCT is the two way coverage chart. The two way coverage chart indicates for any two features if a verification or regression test exists that uses both of those features simultaneously. It can be impractical to add a verification test every possible feature combination. However, the two way coverage report can be used to see if certain key feature combinations are tested together, such as damping in a transient analysis or strain output on shell elements. Lack of a two way coverage test may indicate additional verification testing is needed, though engineering judgment must be applied to identify the most critical feature combinations.

The third FCT output is a of list the top 5 verification tests nearest to (in the sense of using the same capabilities) as used in the input file. If an analysis has a very closely matching rigorous verification test is gives high confidence that the entire use case of the analysis and all feature combinations used are well verified in conjunction.

DISTRIBUTION:

1	MS 0380	M. J. Skroch, 1542
1	MS 0899	Technical Library, 9536 (electronic copy)

v1.40

Sandia National Laboratories