Google Test

From Wikipedia, the free encyclopedia

Google Test is a unit testing library for the C++ programming language, based on the xUnit architecture.[1] The library is released under the BSD 3-clause license.[2] It can be compiled for a variety of POSIX and Windows platforms, allowing unit-testing of C sources as well as C++ with minimal source modification. The tests themselves could be run one at a time, or even be called to run all at once. This makes the debugging process very specific and caters to the need of many programmers and coders alike.

Google Test

Developer(s)	Google Inc.
Stable release	1.8 / August 22, 2016
Development status	Active
Written in	C++
Operating system	Linux, Microsoft Windows, macOS
Type	Unit testing tool
License	BSD 3-clauses
Website	github.com/google/googletest/ (https://github.com/google/googletest/)

Contents

Types of Google Tests

Google introduced its own classification of test types.[3] Instead of categorizations like unit, functional or integration test they have only three categories: small, medium, and large scale tests:

Small Tests (Unit Tests)

Small tests are mostly (but not always) automated and exercise the code within a single function or module. The focus is on typical functional issues, data corruption, error conditions, and off-by-one mistakes. Small tests are of short duration, usually running in seconds or less. They are most likely written by a software engineer (SWE), less often by a software engineer in test (SET), and hardly ever by a test engineer (TE).[4] Small tests generally require sample environments to run and be tested in. Software developers rarely write small tests but might run them when they are trying to diagnose a particular failure. The question a small test attempts to answer is, “Does this code do what it is supposed to do?”[4]

Medium Tests (Integration Tests)

Medium tests are usually automated and involve two or more interacting features. The focus is on testing the interaction between features that call each other or interact directly.[4] Software engineers drive the development of these tests early in the product cycle as individual features are completed and developers are heavily involved in writing, debugging, and maintaining the actual tests. If a medium test fails or breaks, the developer takes care of it autonomously. The question a medium test attempts to answer is, “Does a set of near adjacent functions operate with each other the way they are supposed to?”[4]

Large Tests (Acceptance Tests)

Large tests cover three or more features and represent real user scenarios with real user data sources. There is some concern with overall integration of the features, but large tests tend to be more results-driven, checking that the software satisfies user needs. All three roles are involved in writing large tests and everything from automation to exploratory testing can be the vehicle to accomplish them. The question a large test attempts to answer is, “Does the product operate the way a user would expect and produce the desired results?”[5]

Fidelity

This test will fail under the condition that the code has failed during a test case or the test process itself. In other words; when the code breaks, the test is failed.[6]

Resilience

The test should not fail if a test case does not pass. The test only fails when there is a breaking change that is implemented to the code being tested.[6]

Precision

When the test fails, the exact error will be located and notified to the tester. This type of test does not work well with functions that rely on string inputs.[6]

Others

- Basic Test [1]
- Floating Point Comparisons [1]
- Death Test [1]

Projects using Google Test

Besides being developed and used at Google, many other projects implement Google Test as well:

- Chromium projects (behind the Chrome browser and Chrome OS)
- LLVM compiler
- Protocol Buffers (Google's data interchange format)
- OpenCV computer vision library
- Gromacs molecular dynamics simulation package[7]

Google Test UI is test runner that runs one's test binary, allows one to track its progress via a progress bar, and displays a list of test failures. Clicking on one shows failure text. Google Test UI is written in C#.[8] Additionally, a feature-complete Visual Studio extension exists with Google Test Adapter.[9]

Popular Tools used in Chrome with association to Google Test:

- Developer tools: Look at the page’s elements, resources, and scripts, and enable resource tracking.
- JavaScript Console: Does JavaScript in the Console execute correctly?
- Viewing source code: Is it easy to read through the use of color coding and other help, and is it easy to get to a relevant section?
- Task Manager: Do processes show up correctly and is it easy to see how many resources the web page consumes?[10]

Fixtures

Fixture testing is crucial in computer code because it allows the testing of time and memory management.[1] If these areas are lacking, bugs may arise, and ultimately the code may become incompatible or even fail to run in the first place. Google Test can specifically handle and run this type of test. When doing so, it can also recognize the type of fixture test required. Fixtures in Google Tests are

considered a class and can be instantiated as one as well. Here are some of the details relevant to understanding how fixtures work:

- One could do initialization or allocation of resources in either the constructor or the SetUp method. The choice is left to you, the user.
- One could do deallocation of resources in TearDown or the destructor routine. However, if you want exception handling you must do it only in the TearDown code because throwing an exception from the destructor results in undefined behavior.
- The Google assertion macros may throw exceptions in platforms where they are enabled in future releases. Therefore, it's a good idea to use assertion macros in the TearDown code for better maintenance.
- The same test fixture is not used across multiple tests. For every new unit test, the framework creates a new test fixture. So in Listing 14, the SetUp (please use proper spelling here) routine is called twice because two myFixture1 objects are created.[1]

See also

- List of unit testing frameworks
- CppUnit

References

1. A quick introduction to the Google C++ Testing Framework (http://www.ibm.com/developerworks/aix/library/au-googletestingframework.html), Arpan Sen, IBM DeveloperWorks, 2010-05-11, retrieved 2016-04-12
2. Google Test's repository (https://github.com/google/googletest), retrieved 2016-04-12, cites New BSD (http://opensource.org/licenses/BSD-3-Clause) as license. The license file is at github.com/google/googletest/blob/master/googletest/LICENSE (https://github.com/google/googletest/blob/master/googletest/LICENSE)
3. Test Sizes (http://googletesting.blogspot.ch/2010/12/test-sizes.html), retrieved 2015-04-16
4. Whittaker 2012, p. 12.
5. Whittaker 2012, p. 13.
6. The Google Test and Development Environment (http://googletesting.blogspot.com/2014/01/the-google-test-and-development_21.html), Anthony Vallone, 2014-01-21, retrieved 201-05-10
7. Gromacs Testing Framework (http://manual.gromacs.org/documentation/5.1/dev-manual/testutils.html#unit-testing-framework)
8. Google Test UI (https://github.com/ospector/gtest-gbar) retrieved 2016-04-12
9. [1] (https://github.com/csoltenborn/GoogleTestAdapter)

10. Whittaker 2012, p. 251.

Further reading

- Whittaker, James (2012). *How Google Tests Software*. Boston, Massachusetts: Pearson Education. ISBN 0-321-80302-7.

External links

- Google Test (https://github.com/google/googletest)
- Google Test Primer documentation (https://github.com/google/googletest/blob/master/googletest/docs/Primer.md)
- A quick introduction to the Google C++ Testing Framework (http://www.ibm.com/developerworks/aix/library/au-googletestingframework.html), Arpan Sen, IBM DeveloperWorks, 2010-05-11
- The Google Test and Development Environment (http://googletesting.blogspot.com/2014/01/the-google-test-and-development_21.html), Anthony Vallone, 2014-01-21