International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

Unit Testing Frameworks: A comparative study

Gurubasava[1], Anusha L S[2]

[1]*Student, RV college of Engineering, Karnataka India*
[2]*Professor, RV College of Engineering, Karnataka India*

Abstract - Nowadays the need of software development in every field increased exponentially, thereby increase the need to perform intense software testing. Unit testing plays a major role for a successful software development process. To meet the increased need for producing high-quality software more quickly, or "Quality at Speed," software testing must be carried out more quickly and successfully. The main goal of unit testing is to confirm that each piece of software code operates as intended. This paper is an effort towards comparative study of different unit testing frameworks for various modern languages such as JAVA, C++, and PYTHON.

***Key Words*: – Unit testing, Google Test, Google Mock, TestNG, Junit, CppUnit, CppUTest, Pytest, unitttest, junit and jmock.**

1.INTRODUCTION

Unit testing frameworks are software tools that help in writing and executing unit tests. Provides a framework for building tests as well as the ability to run the tests and report the results. Frameworks can be used as development tools on par with preprocessors and debuggers, therefore frameworks are not just tools for testing. These frameworks are almost contributed at every stage of the development of software including architecture and design of the software, assurance of software quality and performance tuning.

Unit testing is a component of the testing methodologies where the tester is aware of the internal implementation of the proposed software which is being tested. The main purpose of unit tests is to verify the functionality of your code units. Both manual and automated unit testing are possible. In-code manual tests are typically used for small projects where the expense of establishing a testing framework is not required. Using testing frameworks, automated testing is carried out. Pytest is a Python programming language package for automated unit testing.

In this paper a comparative study of the different languages unit testing framework is discussed.

2. STUDY OF VARIOUS LANGUAGES UNIT TESTINGFRAMEWORKS

C++: Chosen some of the most used unit testing frameworks such as CppUnit, Google Test and gMock, CppUTest and Unity for C++ unit testing. And comparative study of each unit testing framework is discussed with different papers as shown in Table I.

TABLE I

Various languages unit testing frameworks

Languages	**Variousunit testing frameworks**
C++	Google Test and Google Mock, Unity, CppUTest and CppUnit
JAVA	Junit and JMock, TestNG,
PYTHON	Pytest, unitest, Doctest

CppUnit: One of the most used C++ unit testing frameworks. CppUnit uses the fixtures, suites and macros also. CppUnit includes a Test Suite class that allows programmer to run many numbers of Testcases concurrently. If having a Test suite for the any code, CppUnit includes tools for defining and displaying the suite's results. A static method suite which returns a test suite that makes the test suite accessible to a TestRunner programme. CppUnit uses For executing testcases first required to define the path with using ($TargetPath) in Visual Studio Code there is no direct way to run it on any of the IDE's. For checking the output of the function for which unit testing is wrote by programmer. CPPUNIT_ASSERT is used for checking the function output is coming exactly as expected or not.

CppUnit has many disadvantages one of among is to have the own test case, must create a subclass [1]. Inheritance is a property which connects the strong bonding between two classes. The requirement that a test-case class inherit from a CPPUnit-framework class binds the two together.

Google test (Gtest) and Google mock (gMock):

Gtest: Gtest is open-source unit testing framework developed by google which is helpful for writing unit testcases for each function and running the tests for C++ coding language. In terms of compatibility, it is used in many operating systems such as Windows, Mac, and Linux and it supports both Bazel and CMake for building [2].

Figure 1 above shown is the basic methodology to write a unit test case.

Start the testcase with macro–TEST. Inside that TEST macro write the code responsible for the test execution

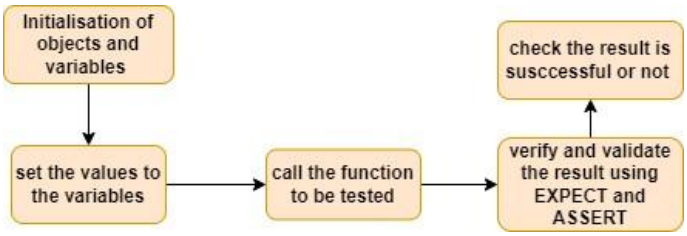

Fig. 1. Unit testcase writing methodology

requirements such as initialization of variables and objects, setting the values to those variables and calling that function need to be checked and check the results by using the macro's EXPECT and ASSERT. The major difference between them is ASSERT stops the execution if the test is unsuccessful [2] whereas EXPECT macro continues its execution till the code ends.

gMock: gMock is also a mocking library developed by google itself. The main goal of the gMock is to provide a mechanism for implementing mock classes that mimic a portion of the tested system within googltest tests. Its operating principle is based on the interaction between tested code and the mock objects. These objects have same interface as the system component that they replace. They use the same abstract class as the actual component. Figure 2 below is the basic highlight for the mocking the function.

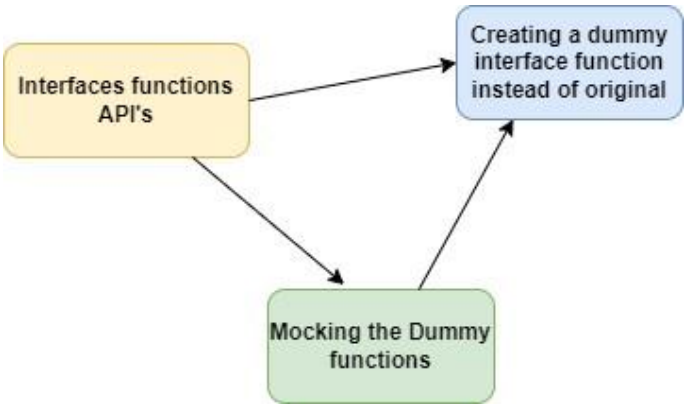

Fig. 2. Function mocking methodology

MOCK_METHOD is a macro utilized inside the mock class. There are things to declared inside the macro and they are

1. Name of the real method used
2. Return type
3. Arguments of the real method name used in mock class [2]

Here EXPECT_CALL macro also plays a major role for setting expectations to the functions such as how many times function to be invoked, how may time the function should return the value [2] etc.

CppUTest: CppUTest is a test framework for unit testing primarily used for embedded systems that supports both the C and C++ programming languages. This framework was written in C++, but test cases can be written in C without knowing C++. This makes it easier for C programmers to work with this framework. It is similar to Unity except that it does not require a Test Runner. As a result, it overcomes Unity's main disadvantage [3]. As a result, it reduces the amount of work and time required to update test cases in the test runner. It supports multiple operating systems and is primarily intended for use in embedded software development [4].

The setup () and teardown () routines must be specified inside the TEST_GROUP (Group Name) in the CppUTest Test, even though each group includes several test cases, one setup, and one takedown. In terms of framework organization, CppUTest and Unity diverge a little. Due to the fact that everything is described inside the group, it appears more compact than Unity. The testcase format is as follows in Figure 3.

```
TEST_GROUP (group name)
{
        define set up () function
        define tear down () function
         //use helper functions if required
}
```

Fig. 3. Basic code example

The result of the function is working as expected or not is checked by using the below listed macros TABLE II.

TABLE II

Result checking macros list

Result Checking macros	Explanation
CHECK(Boolean)	Returns true or false
CHECK_TRUE(bool condition)	It returns successful, if condition is true
CHECK_EQUAL(actual, expected)	Checks whether actual and expected are equal or not
STRCMP_EQUAL	It will compare two strings

Comparison of C++ unit testing frameworks

Compared to Google test and CppUnit, for CppUTest, the test runner, is not required, which saves a tonne of time and effort. This framework also adheres to the four-phase test pattern, which contributes to well-organized code that is simple to read and comprehend [4]. Mocking is option is not there for both CppUnit and CppUTest compared to Gmock and setting the expectations is also possible for gMock library unit testing framework [2]. Google Test consists of various kind of macros options, and it consists of more features to execute the test results easily when compared to both CppUnit and CppUTest.

JAVA:

JUNIT and JMOCK:

JUnit: Junit is one of the best and most widely unit testing framework used across the entire IT sector for java programming. It is a open source unit testing framework. Programmer stress and debugging time are decreased as a result of the increased productivity of the programmer and the stability of the programme code.

Jester is used as tester for running JUnit tests; it alters the source code in various ways and determines whether the tests are still passable after each change. Jester identifies possible code modifications that can be made without causing the tests to fail. If changes to the code can be made without the tests failing, either a test is missing, or the code is redundant [5]. Jester can be used to determine whether the current tests are sufficient or to provide information on the tests that are lacking.

JUnit has design rules for writing and executing the unit testcases. In [6], explained in detailly about how the class of TestClass is derived from junit and collaborating with the target class explained [6].

JMock: JMock offers an easy-to-use and expressive API for mimicking interfaces, defining anticipated invocations, and executing invoked behavior [7].

Writing tests that can be executed and read as documentation is a unique purpose of jMock. The majority of the jMock API is devoted to creating readable syntactic sugar for expectations definition. This ambition, which aims to develop a domain-specific embedded language [7] hosted in Java, has resulted in an API that is extremely unorthodox when compared to typical Java designs.

Even though jMock has a huge library of restrictions and matching criteria, it is not able to accommodate every case that a programmer would require. In fact, adding constraints that are unique to your problem domain makes your tests clearer, proving that matching rules and constraints are extendable.

In order to make it simple to determine what led to a test failure, jMock is made to give helpful messages. Mock objects have names that make it simple for the programmer to connect error messages to how the test was executed and the target code. A clear failure message can be produced by combining descriptions from the core objects that are used to express expectations.

TestNG: TestNG is one of the most popular open-source testing frameworks for automation testing for java. TestNG framework is developed for any web application development along with using selenium webdriver [8].

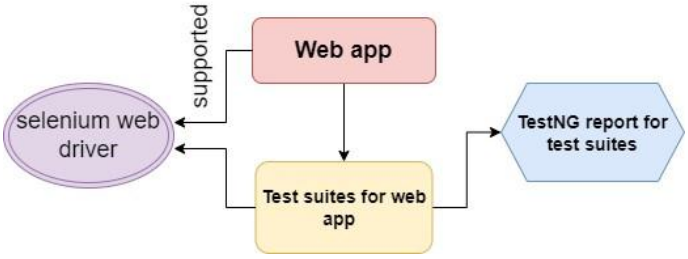

Fig. 4. TestNG with selenium webdriver

In [8], selenium webdriver supports the web application and test suites written for the web application are also supported by the selenium webdriver which uses the test suites written for the web application using TestNG framework and which is automated by using selenium webdriver. So that its pipeline directly it will generate report based on the test suites written by the developer with selenium webdriver and a third-party tool produces a screenshot of the report [8].

Comparison of TestTNG and Junit

TestNG offers certain new features that give it an edge over JUnit in terms of power. All test types, including unit, functional, and integration testing, are covered by TestNG. JUnit cannot able to run parallel tests, won't support advance annotations, grouping tests is not possible and dependency tests are also not executable whereas all these features are supported by TestNG framework.

Python

Pytest and unittest

Python developers are majorly using pytest and unittest two-unit test framework to test their python code. Nowadays python projects are migrating from unittest to pytest there are some specific reasons which discussed in paper [9].

Several reasons are there for migrating from unittest to pytest those are easier syntax defined, fixture reuse, easy maintenance and implicit mechanics of the pytest major reasons of the migration [9].

In paper [9], survey is conducted upon migration of testing in from unittest to pytest which concludes that 66% of initial developers rely on unittest now converted to pytest because of various advantages over unittest.

Migrating from unittest to pytest involves following steps

1. Removing the test defined inside the class and migrating to the normal functions.
2. Updating the normal assertions to asserts in pytest.
3. Replacing setup function operations to fixture

Example of the sample code in pytest and unit test as shown in Figure 5.

Pytest	unittest
def test_do(): assert False	From unittest import TestCase class DoTestClass(TestCase): def test_do(self) self.assertFalse(False)

Fig. 5. Sample code of pytest and unittest

Migration from unittest to pytest is become quite common because the survey conducted by the [9] proves this. Migration to pytest also have many advantages and disadvantages over unitest and future work in this field covered [9]. To migrate the testing framework is also possible form unittest to pytest[9].

Comparison of pytest and unittest

When compared to unittest, Pytest has many inbuilt capabilities that need less code. For unittest, we first need to import a module, then we need to make a class, and last we need to specify testing functions inside the class. However, in the case of Pytest, we must specify the functions and make the criteria included within them explicit. Overall, the format of unittest use cases is difficult, there is no compatibility, there are few plug-ins, and secondary development is easy. The use case format is straightforward, and Pytest is more practical and quicker. It offers strong compatibility and can run unittest-style test cases without altering their source code. Pytest plug-ins are many and include the flash plug-in, which can be used to rerun tests in the event of an error, and the xdist plug-in, which allows for more productive parallel device execution.

3.CROSS COMPARISON BETWEEN DIFFERENT LANGUAGES UNIT TESTING FRAMEWORKS

When compared to java, C++ doesn't have certain of Java's reflection features, we must create our own main function for testing. For straightforward situations, we merely create a runner object and allow our test to run. Despite the complexity of the language calling for it, C++ programmers lacked the tool support for simple-to-use unit testing. Without tests, refactoring and streamlining C++ code is extremely challenging and error prone. So java unit testing unit testing frameworks are better in comparison with c++ unit testing frameworks.

In case of selenium, No matter the technology, it is used to automate web application testing. Due to its simplicity of setup, Python may be the best choice for Selenium automated testing. Python is frequently preferred over Java by start-ups and medium-sized businesses because of its straightforward programming syntax. Python makes it considerably simpler to develop Selenium programmes than Java does. In addition, the Python Selenium framework PyTest makes it the ideal choice to take advantage of the creation of sophisticated functional tests.

4.CONCLUSIONS AND FUTURE WORK

We have discussed about various languages unit testing frameworks such as C++, JAVA and PYTHON etc. Here if we compare among the C++ available unit frameworks googletest and google mock are best resource since they are open source also acquired many features when compared to other frameworks. Whereas in case of JAVA Junit is the best when compared to jmock and TestNG. In case of PYTHON, pytest is the best because in paper [9] we have seen the migration from unittest to python.

In future work, we have to see for such kind of framework or tool if we dump the any languages developed code for testing means automatically tool should test everything without depending on any technology it should be technology independent.

ACKNOWLEDGEMENT

The authors thank ANUSHA L S for the collaboration of preparing this paper and Zebra technologies for providing opportunity to work in this area.

REFERENCES

[1] Peter Sommerlad and Emanuel Graf. "Cute: C++ unit testing easier". In: *Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*. 2007, pp. 783–784.

[2] Nenad Petrovic et al. "Model-driven automated gMock´ test generation for automotive software industry". In: *XV International SAUM* (2021), pp. 1–4.

[3] Lari-Matias Orjala. "Unit testing methods for internet of things Mbed OS operating system". In: *University of Oulu, PhD diss* (2019).

[4] Sandhya Mudduluru. *Investigation of Test-Driven Development based on Mock Objects for Non-OO Languages*. 2012.

[5] Ivan Moore. "Jester-a JUnit test tester". In: *Proc. of 2nd XP* (2001), pp. 84–87.

[6] Michael Wick, Daniel Stevenson, and Paul Wagner. "Using testing and JUnit across the curriculum". In: *ACM SIGCSE Bulletin* 37.1 (2005), pp. 236–240.

[7] Steve Freeman et al. "jMock: supporting responsibility based design with mock objects". In: *Companion to the 19th annual ACM SIGPLAN conference on*

Object-oriented programming systems, languages, and applications. 2004, pp. 4–5.

[8] Satish Gojare, Rahul Joshi, and Dhanashree Gaigaware. "Analysis and design of selenium webdriver automation testing framework". In: *Procedia Computer Science* 50 (2015), pp. 341–346.

[9] Lıvia Barbosa and Andre Hora. "How and Why Developers Migrate Python Tests". In: ().