International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume: 3 | Issue: 2 | Jan-Feb 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 - 6470

Unit Testing to Support Reusable for Component-Based Software Engineering

Khin Moe Sam

University of Computer Studies, Thaton, Mon State; Myanmar

ABSTRACT
Unit testing is a practical approach to improve the quality and reliability of software. Unit testing is usually performed by programmers and is the base for all other tests such as integration testing and system testing. Unit Testing can be done manually (and/or) automatically. The automated unit tests are written by the developers after the completion of functionality coding. The number of defects reduced when automated unit tests are written iteratively similar to test driven development. This framework proved that significant portions of windows application can be automatically tested without manual intervention. This reduces the manpower involved in testing each and every unit of the application and increases the quality of the software product.

Keywords: *Unit Testing, Automatic Unit Testing, And Test Driven Development.*

1. INTRODUCTION
Unit Testing is a kind of white box testing where individual units of software are tested. The intension of unit testing is to check whether each and every unit (module) of a software works as per the developer's expectation. Unit Testing can be done manually (and/or) automatically. Unit Testing finds the defects in each and every unit of the application at initial level of testing. It increases the confidence levels of developer by ensuring that their code is working correctly. Unit Testing ensures that code works correctly and improves the quality, reliability and cost of the application software development. We initially performed unit testing manually where it was a time consuming task and hence we intend to choose automation of unit testing. To automate unit testing, it is necessary to write test scripts called unit tests. Once the unit tests are available, it is easy to automate the unit testing. (1) The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. A combination of the two or more tools can serve to merge the best of both worlds. The result is not only the sum of both mechanisms, but the creation of a new quality of testing tool with new functionality emerging out of the combination of the two.(3)

2. Objectives
The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. (4)A combination of the two can serve to merge the best of both worlds. The result is not only the sum of both mechanisms, but the creation of a new quality of testing tool with new functionality emerging out of the combination of the two. (3)(5)Unit testing is performed on the smallest elements of a system; each component is tested to ensure that it properly works. Usually it performs a single cohesive function. The goal of unit testing is to analyze each small part of the code and test that is working correctly. (6) The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. (2)Benefits of writing good unit tests or unit testing:- Unit testing supplies plentiful benefits: finding software bugs early, simplifying integration, providing a source of documentation, and many others, which I am going to share in more detail: Unit testing reduces the number of bugs in the system and acceptance testing; Cost of locating and fixing bugs in unit testing is very less as they are captured in very early phase; Automated tests can run as frequently as required; Unit testing makes it easier to change and refactor the code; Unit testing can improve code design, especially with test-driven development; Overall development time can be used for unit testing; Unit tests are a form of documentation; Unit testing improves teamwork; Code coverage can be measured.(7)

3. Methodology
Automated unit testing helps in improving quality, decreasing costs and reducing time required for testing. However, all this is possible only with use of appropriate testing tool. Following are various parameters critical to selection of appropriate tool: Minimum implementation time: This is possible when users are already familiar with technology (language, method of working, integration constraints) used by the testing tool. Additionally, extensive documentation and support available help in reducing implementation time. Minimum ownership and running costs: An open source tool which is free to access may be less efficient and have higher operating costs. On the other hand, a commercial tool available may require initial investment but due to its better features might make testing efficient and cost effective in the long run**.** Flexibility**:** Every project is unique in a certain way and has its own peculiar needs. Thus an efficient testing tool while offering most required features should also provide option of code modification. Further for swift debugging, the framework should make test code readable. (8)

The automated unit tests for any application can be written in two ways: Before code implementation; after code implementation. If unit tests are written before any code implementation then it is known as Test Driven Development. If Unit tests are written after the code implementation then it is known as Test after Development. (9) The best unit testing tools are NUnit, TestNG, JUnit, JMockit, Emma, Quilt, HtmlUnit. JUnit, TestNG, NUnit are free open source unit testing tools. Nunit is a unit testing framework based on .net platform. It is free tool allows to write test scripts manually but not automatically. It works in the same way as JUnit works for Java.

It supports data-driven tests that can run in parallel. It uses console runner to load and execute tests. JUnit is an open-source unit testing framework designed for Java Programming language. It supports for test-driven environment and the core idea on which it is based 'first testing then coding'. Test data is first tested and then inserted in the piece of code. It provides annotation for test method identification, assertion for testing expected result and test runners. It is simplest and helps to write code easily and faster. TestNG is an open-source automation testing framework for java programming language. This tool is heavily influenced by JUint and NUnit with concurrent testing annotation support. TestNG supports parameterized and data-driven testing along with unit functional and integration testing. It proved effective with powerful execution model and flexible test configuration.

4. Proposed System

In Test Driven Development, the developer writes automated unit tests for the new functionality they are about to implement. It is a software engineering process that follows small development cycle. In industry, while coding a software application the development cycle that they follow is shown below in figure 1.

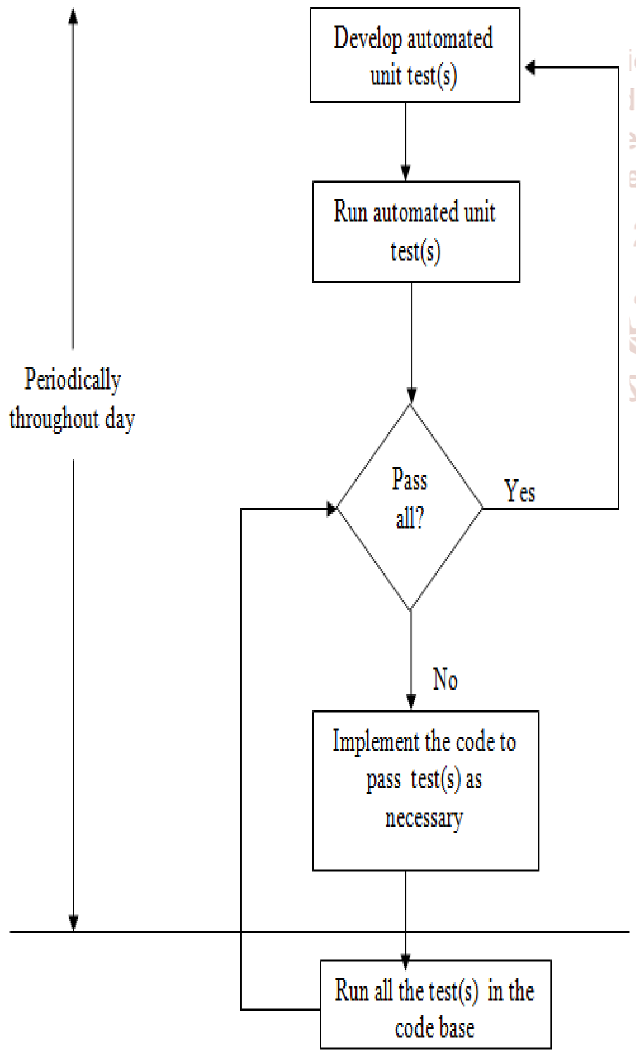
Develop automated unit test(s)
Run automated unit test(s)
Pass all?
Yes
No
Implement the code to pass test(s) as necessary
Run all the test(s) in the code base
Periodically throughout day

Fig.1. Test Driven Development Cycle

4.1 Develop Automated Unit Tests

Without writing any code to implement feature, test cases must be written by the developer in the initial stage by collecting the specifications and requirements in the form of user stories (or) use cases that covers all the requirements and conditions. This makes developer to focus on requirements and coding in this manner makes the code consistent.

4.2 Run Automated Unit Tests

Run the automated Unit tests to ensure that they fail because there is no implemented code yet.

4.3 Implement the Code to Pass Test(s)

Developer needs to write the code for those cases that are failed in the previous test. The code that developer writes should not add any other unpredicted functionality.

4.4 Run All the Tests in Code Base

Once development is done, run all the automated tests in the code base. If all the tests are passing then develop automated unit tests for other features of the application and repeat the same process otherwise implement the code necessary to make the test pass and run the automated unit test(s). Finally, once the development of code for the application and unit testing are done, the developer may restructure the code for better readability (or) improving the performance. The advantage of above written unit tests is that whatever changes the developer may make to the code now, it won't affect the existing functionality of the build, as the test cases written earlier defines the requirement and specifications of the application.

5. Conclusion

Unit tests isolate each part of the program and check that the individual parts are correct. Unit testing finds problems early in the development cycle and hence facilitates changes required in code. Testing the parts of a program first and then testing the sum, allows faster integration testing and progressive documentation which are critical to the success of the unit. However, considering the variety of testing tools available and their variable features, it becomes essential that features of each testing tool are analyzed in detail before any selection and use. As a future work direction, we plan to apply mutation testing and analysis to the test suites generative by our tool to assess their fault detection effectiveness and to compare that effectiveness with those of the test suites generated by other comparable tools. We also plan to compare the effort and time spent in using those tools. Also, the execution time of our tool to automatically generate test case code is very fast, and we plan to compare the associated efficiency to manual coding when our tool is used in large-scale projects. (10)

References

[1] Aswatha Kumar M. et al. (Eds.): Proceedings of ICAdC, AISC 174, pp. 813–822. springerlink.com © springer India 2013)

[2] Software testing fundamentals.com/ unit-testing/(ISTQB)

[3] Andrew Patterson; Faculty of Information Technology; Monash University; Australia; Introducing Unit Testing With BlueJ,ajp@infotech. monash.edu.au

[4] Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. p. 75. ISBN 0-470-04212-5.)

[5] johnr@infotech.monash.edu.au//

[6] *https://apiumhub.com/tech-blog-barcelona/top-benefits-of-unit-testing/*

[7] By Ekaterina Novoseltseva; Jan. 18, 17 · DevOps Zone; http://crestechglobal.com/the-importance-of-unit-testing-in-software-testing/

[8] By Arvind Rongala, Manager, Business Development and Marketing, Invensis Technologies) ;March 28, 2015

[9] A.N. Seshu Kumar and S. Vasavi; Effective Unit Testing Framework for Automation of Windows Applications; V.R. Siddhartha Engineering College, Vijayawada {seshu1203, vasavi.movva} @gmail.com

[10] Christian Wiederseiner1, Shahnewaz A. Jolly1, Vahid Garousi1, and Matt M. skandar; An Open-Source Tool for Automated Generation of Black-Box xUnit Test Code and Its Industrial Evaluation; Software Quality Engineering Research Group (SoftQual), University of Calgary, Canada; MR Control Systems International Inc., Calgary, Canada, {christian. wiederseiner, sajolly,vgarousi} @ucalgary .ca,matt. eskandar@mrcsi.com)