Differentiating Integration Testing and Unit Testing

Hanmeet Kaur Brar
UIET, Panjab University
Chandigarh, INDIA
Email Id: brar.hmnc@yahoo.com

Puneet Jai Kaur
Asstt. Professor, UIET, Panjab University
Chandigarh, INDIA
Email Id: puneet@pu.ac.in

Abstract – Software testing plays a vital role in the affirmation of the software quality and reliability. It is a never ending process which starts with the very initial stage of software development and continues with Software Development Life Cycle. Testing may be in any form depending on the type of software under consideration. This paper furnishes the basics of Software Testing and steps involved in the testing procedure. We will review two significant approaches for testing software which are Unit testing and Integration Testing. We will study the brief overview of the both the approaches and also compare them on the basis of various parameters like modules, problem discovery and techniques involved.

Keywords-integration testing, software development life cycle, software testing, software quality, unit testing.

I. INTRODUCTION

Testing is an essential and continuous process in Software Development. Software quality and reliability highly depends on testing it has undergone. It is a costly as well as time consuming process [1]. About 30% to 60% of the development cost of software is constituted by testing [2]. But it is better to invest in testing rather than spending a lot more on facing the damages at later stages when sometimes the damage is irreversible. Various approaches are available for testing software and can be adopted according to the requirement. A tester opts for different testing technique at different levels of Software development.

Different levels of software testing may be categorized as, Unit testing, Integration testing, Functional/System testing, Acceptance testing. Unit testing tests the small units individually. Integration testing tests the integrated versions of the units to examine their interactions. The whole system is tested for its functionality after the integration of all the units and this approach is called System testing. And in Acceptance testing, the software is tested to examine its adherence to the requirements specified by the customer.[3]. These levels do not have any definite and strict demarcations from each other.

Section II of this paper gives introduction of Software testing. Section III presents description of Unit testing with Section IV presenting overview of Integration testing and differentiating both Unit testing and Integration Testing in Section V. Finally, Conclusion and Future Work are in Section VI and VII respectively.

II. TESTING

Software testing may be defined as the technique to examine the software for defects. These defects may be non-conformance to the user requirements, some problem with the logic, implementation issues or human error. It verifies and validates the software which means it determines that the software produced (or being produced) is correct as well as determines if correct software is produced (or being produced).

Software testing plays a vital role in the affirmation of the software quality. Various objectives of software testing include uncovering the defects as well as preventing them to propagate further, providing genuine software which meets the demands of the customer, and diminishing the creation cost as well as the maintenance cost [4]. It is a never ending process which starts with the very initial stage of software development and continues in whole Software Development Life Cycle. Earlier the testing is started, fewer are the defects proliferated.

The approaches used for Software testing may be broadly categorized as: White box testing and Black box testing. White box testing mainly involves dealing with the code and is required to be done by the developer himself. Whereas Black box testing is concerned with the checking of the software without code being involved.

Different steps involved in the testing procedure are:

1. Analysis of user requirements
2. Generating the test cases
3. Carrying out the tests by executing the test cases
4. Outlining the test results
5. Analysing the results and acting for betterment. [5]

III. UNIT TESTING

Unit testing means testing the small functional units of the project individually to check if they are performing the way they are expected to behave by executing all the possible paths. The generation of test cases, execution of test and study of results employ great technical skills. So this test is usually done by the developer himself who has complete knowledge

978-9-3805-4416-8/15/$31.00 ©2015 IEEE

of the code. This test is done to examine the basic logic. Time taken for the execution of Unit Test is dependent upon the size of the unit under examination.

This is the very root level approach towards the quality enhancement due to the reason that if individual components are tested and made error free or improved upon quality, then this results in less no of errors after being combined to form a bigger system and hence more dependable software is developed.

This depends upon the tester (ie. Coder in this case) that what amount of code is considered as the unit to be fed as input to Unit test. And this may vary from person to person. Unit testing may begin as soon as a unit is coded.

Most of the people believe that there are various issues with Unit testing like it employs great deal of time as well as capital. But these things depend on the applications to be tested.

IV. INTEGRATION TESTING

Integration testing examines if the individual components developed and tested separately, interact as expected when they are combined to form a part of larger system. It includes White box testing as well as Black box testing and may be performed by the coder himself or the person with the knowledge of the design of the software/system.

Integration testing may be performed in 4 different ways:

- Top down: Testing the units at high level first and then testing the units at lower levels. In case a unit of lower level is missing and is required for the testing of its higher level unit, then the place of missing unit is given to a 'stub' temporarily which is a simpler dummy unit providing the desired interfacing data flow. The count of 'stubs' those are required to be developed for integration technique, constitute to testing effort. So, reducing the count of 'stubs', also diminishes the cost of integration [6].

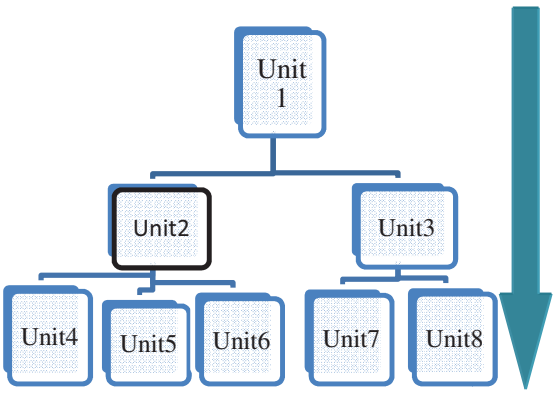

Fig. 1. Top Down Approach of Integration Testing

Here if U1 is to be tested and U2 is yet to be developed , then U2 will be replaced by a 'stub'.

- Bottom up: Testing the units at lower level with the help of the higher level units and this process continues till all the units are tested [7]. Sometimes 'drivers' are required to provide the invocation or the input or to accept the output in case any required module is yet to be developed or is unavailable at the testing time [8].

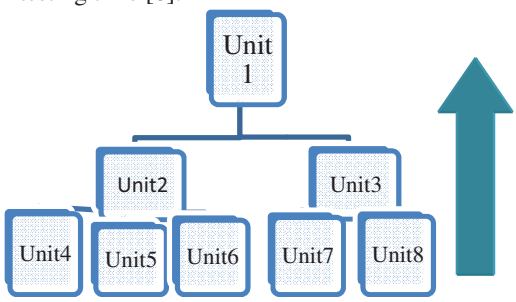

Fig. 2. Bottom Up approach for Integration Testing

- Big Bang: After the small units are tested separately, they are all united at once and then tested after such integration [8]. This approach is defined as Big-Bang approach for integration testing .

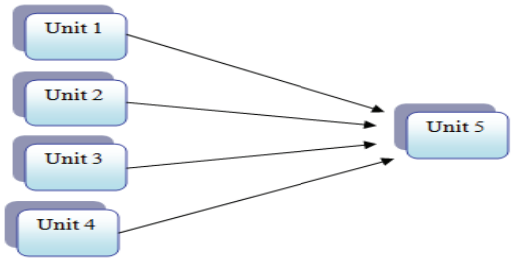

Fig. 3. Big Bang Approach for Integration Testing

- Sandwich: It is the mixed approach which combines both top-down approach as well as bottom-up approaches for integration testing. For more effective testing and to keep the testing procedure simple, Incremental approach is used in Integration testing where the module is tested after each small addition. [5].

Sometimes, it becomes very complex to develop test cases and maintain them in case of Integration Testing [9]. This worsens the issue as the size of the application increases.

V. DIFFERENCES

- In Unit testing, the small functional units of the project are individually examined to check if they are performing the way they are expected, by executing all the possible paths. While in Integration testing, the components formed by uniting two or more units are checked for their interaction with each other.
- Generally the concentration is on a definite functionality to be implemented in case of Unit testing. And Integration testing covers the various functionalities or interactions

which are the result of integration of various dependent or independent functionalities.

- The problems are easy to discover in case of Unit test from their results. This means the problem area is uncovered using Unit testing but Integration testing is not much capable of discovering the problems. This is not certain that the problem area or cause is caught by Integration testing. For example, it may tell that certain units are not interacting properly but the reason for the deviation of their behaviour may or may not be discovered.
- Unit testing involves white box testing whereas Integration testing involves black box as well as white box testing [10].
- Unit testing usually does not consider the effect of external components and only the unit to be tested is focussed upon however in Integration testing, the external components and their effects are considered rather this consideration is a vital part of maximum Integration tests[11].
- In case of Unit testing, testing process or test case generation can be done only by a technician who has the full knowledge of the code (usually the coder himself) whereas in Integration testing , the testing can be done by tester with knowledge of the system's architecture or design or it can be done by the coder.
- Unit testing is done to examine the basic logic of the unit or component under examination. Integration testing is done to examine the interaction of various modules or a module and some specific environment.
- Unit testing may begin as soon as a unit is coded. And Integration testing begins when the interactions among the units come into picture.
- In case of failure of a unit test case, there might be problem in one or very few areas but if an integration test case fails, the problem might involve larger no of areas.

TABLE 1. COMPARING UNIT TESTING AND INTEGRATION TESTING

Parameter	Unit Testing	Integration Testing
Modules to be tested	Small functional units	Interaction of various units
Problem discovery	Easy	Difficult
Technique involved	White Box Testing	White Box Testing, Black Box Testing
External Factors	Generally not considered	Considered
Performed by	Coder	Coder or the design architect
In case of test case failure	Problem areas is one or few	Might involve larger no. of areas

VI. CONCLUSION

Testing shows and improves the dependability of software. This paper compares the two main testing approaches namely Unit Testing and Integration testing. Unit testing examines the small functional units of the project individually to check if they are performing the way they are expected, by executing all the possible paths. Whereas, Integration testing, checks for the interaction of the components formed by uniting two or more units with each other. Both the approaches are important stepping stones for quality improvement.

VII. FUTURE WORK

In future, we can try combining both the Unit testing as well as Integration testing techniques to provide better testing environment and improve the efficiency.

As we have discussed that there are few limitations of Unit testing which include employment of large capital and time. We can work on these in future for any particular application to reduce the capital or time required.

Same kind can be done in case of Integration Testing. Work can be done to address the issues like increasing complexity of Integration Testing with increasing size of the application.

REFERENCES

[1] M. Ellims, J. Bridges, D. C. Ince, "The Economics of Unit Testing", *Springer Science Business Media, Inc. 2006, Vol 11, pp.5-31.*

[2] Y.Labiche, "Integration Testing Object-Oriented Software Systems: An Experiment Driven Research Approach", IEEE CCECE 2011 Niagara Falls, Canada,pp. 000652-000655.

[3] G. Saini, K. Rai, "Software Testing techniques for test case generation", *International Journal of advanced Research in Computer Science and Software Engineering,* Vol. 3, Issue 9, pp. 261-265 September2013.

[4] Naresh Chauhan-Software testing principles and practices; Oxford, 2010; pp.3-15, 65-88

[5] S. H. Trivedi, "Software Testing techniques", *International Journal of advanced Research in Computer Science and Software Engineering,* Vol. 2, issue 10, pp. 433-439 , October 2012.

[6] Y. L. Traon, T. Jeron, J-M. Jezequel, and P. Morel, "Efficient Object-Oriented Integration and Regression Testing", *IEEE Transactions on Reliability,* Vol. 49,No. 1, March 2000, pp. 12-25.

[7] https://www2.informatik.huberlin.de/~hs/Lehre/2004-WS_SWQS/20041126_Ex_Integration-testing.pdf

[8] http://www.softwaretestinggenius.com/various-approaches-in-integration-testing

[9] M. Pezze, K. Rubinov, J.Wuttke, "Generating Effective Integration Test Cases from Unit Ones", *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, pp. 11-20.*

[10] I. Ali, "A comparison between Black Box and White Box Testing", *Interesting Results in Computer Science and Engineering, 2009.*

[11] http://stackoverflow.com/questions/5357601/whats-the-difference-between-unit-tests-and-integration-tests