Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 7 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Reducing Testing Effort in the Test Driven Development

By Naveed Khan, Muhammad Shahid Khan, Muhammad Ahmed Javed & Muhammad Abid Khan
Gandhara University, Pakistan

Abstract - Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, and then produces code to pass that test and finally refractors the new code to acceptable standards. TDD is a good approach for the development of the new software but it is more time consuming process model when test the existing software system. In this research we are introducing a new technique which reduces the effort of the TDD approach.

GJCST-C Classification : D.2.5

Strictly as per the compliance and regulations of:

Reducing Testing Effort in the Test Driven Development

Naveed Khan [α], Muhammad Shahid Khan [σ], Muhammad Ahmed Javed [ρ] & Muhammad Abid Khan [ω]

Abstract - Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, and then produces code to pass that test and finally refractors the new code to acceptable standards.

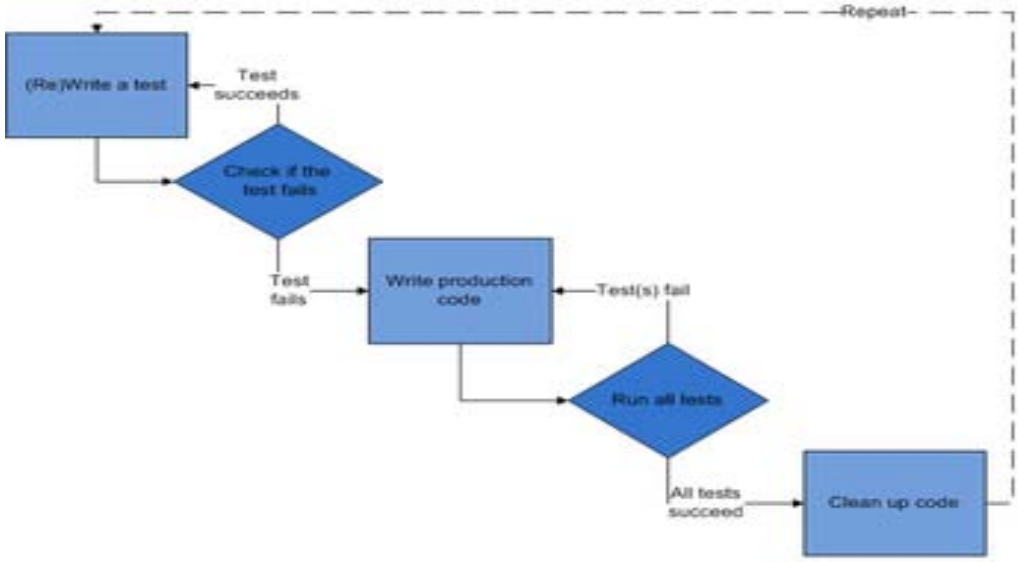
TDD is a good approach for the development of the new software but it is more time consuming process model when test the existing software system. In this research we are introducing a new technique which reduces the effort of the TDD approach.

I. Introduction

Test driven development works on test first programming concept. In test driven development the programmer writes the code and then passes the code through test. If test was successful then stop the process of testing. So pass another code of the project through test if it fails then programmer will modify the code and pass the modified code again through test. Process will repeat again and again until test of the code will be successful.

II. Research Question

The existing test driven development model for unit testing work very fine for the newly software but it is time consuming process for the existing software. In order to reduce the testing effort for the existing software in test driven development we need some improvements. Test Driven Development Model is shown bellow.



In the above model "write a test" and "write production code" a lot of typing effort is required by the programmer so testing effort will be increase if we implement this model for the existing system. Our research question is that how we reduce the testing effort in the test driven development?

III. Key Hypothesis

- For test driven development input a module of the existing system.
- Test for module performance.
- If the code of module is working fine then test is successful.
- If test is unsuccessful then Go to module library.
- So select a specfic module from the module library.
- Repeat the first step and test the selected code of the module of module library.

This research is practically is very important and challenge because in this area only newly software are tested no one work on the existing system.

Authors α σ : Gandhara University, Peshawar, Pakistan.
E-mails : naveediit@gmail.com, shahidkhan123@gmail.com
Author ρ : Government Degree College, Kohat, Pakistan.
E-mail : ahmed.javed725@gmail.com
Author ω : University of Engineering & Technology, Peshawar, Pakistan. E-mail : engrabid08@gmail.com

IV. Literature Review

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards. Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.[1]

Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999,[2] but more recently has created more general interest in its own right.[3]

Programmers also apply the concept to improving and debugging legacy code developed with older techniques.[4]

A 2005 study found that using TDD meant writing more tests and, in turn, programmers who wrote more tests tended to be more productive.[7] Hypotheses relating to code quality and a more direct correlation between TDD and productivity were inconclusive.[8]

Programmers using pure TDD on new ("greenfield") projects report they only rarely feel the need to invoke a debugger. Used in conjunction with a version control system, when tests fail unexpectedly, reverting the code to the last version that passed all tests may often be more productive than debugging.[9]

Test-driven development offers more than just simple validation of correctness, but can also drive the design of a program.[10] By focusing on the test cases first, one must imagine how the functionality will be used by clients (in the first case, the test cases). So, the programmer is concerned with the interface before the implementation. This benefit is complementary to Design by Contract as it approaches code through test cases rather than through mathematical assertions or preconceptions.

Test-driven development offers the ability to take small steps when required. It allows a programmer to focus on the task at hand as the first goal is to make the test pass. Exceptional cases and error handling are not considered initially, and tests to create these extraneous circumstances are implemented separately. Test-driven development ensures in this way that all written code is covered by at least one test. This gives the programming team, and subsequent users, a greater level of confidence in the code.

While it is true that more code is required with TDD than without TDD because of the unit test code, total code implementation time is typically shorter.[11] Large numbers of tests help to limit the number of defects in the code. The early and frequent nature of the testing helps to catch defects early in the development cycle, preventing them from becoming endemic and expensive problems. Eliminating defects early in the process usually avoids lengthy and tedious debugging later in the project.

V. Working of the Proposed Model

a) Steps

- For test driven development input a module of the existing system.
- Test for module performance.
- If the code of module is working fine then test is successful.
- Otherwise test is unsuccessful.
- So select a specfic module from the module library.
- Repeat the first step and test the selected code of the module of module library.

Modules of the existing software are stored in the module library. In this way your test effort will be reduced in the existing system. You just test the module code and attach it to the specfic software.

b) Results

In this research we are testing the developed system not a newly system.

Test#	Time taken by Test Driven Development Model	Time Taken by the Proposed Model
1	5 mints= 300 sec	3sec
2	50 mints= 3000 sec	5sec
3	100 mints= 6000 sec	7 sec
4	150 mints=9000 sec	8 sec
5	200 mints=12000 sec	10sec

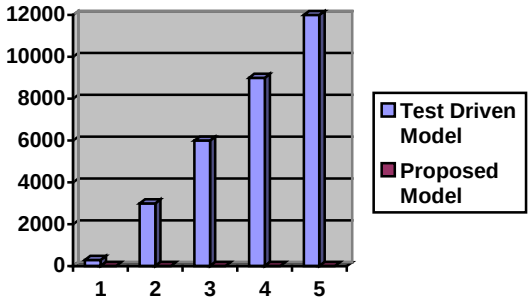


Figure 1.1 : Test cases for both test driven development model and proposed model

VI. Discussion

In the above fig(1.1) shows that proposed model is more efficient then the existing test driven development model. In these tests we write manu lay code in the test driven development model and copy and paste code in the proposed model. Because in test driven development model programmer manually write the code which is time consuming process but in the proposed model we just copy and paste code of the testing module. The above graph shows that the proposed model is 100% reduced the test effort.

VII. Conclusion

The existing test driven development model for unit testing work very fine for the newly software but it is time consuming process for the existing software. In order to reduce the testing effort for the existing software in test driven development we introduced a new approach and practically result shows that the new approach is more efficient for test driven development for the existing systems.

References Références Referencias

1. a b c Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003.
2. Lee Copeland (December 2001). "Extreme Programming". Computer world. Retrieved January 11, 2011.
3. a b Newkirk, JW and Vorontsov, AA. Test-Driven Development in Microsoft .NET, Microsoft Press, 2004.
4. Feathers, M. Working Effectively with Legacy Code, Prentice Hall, 2004.
5. a b c d e f g "Effective TDD for Complex Embedded Systems Whitepaper". Pathfinder Solutions.
6. "Agile Test Driven Development". Agile Sherpa. 2010-08-03. Retrieved 2012-08-14.
7. Koskela, L. "Test Driven: TDD and Acceptance TDD for Java Developers", Manning Publications, 2007.
8. a b "Test-Driven Development for Complex Systems Overview Video". Pathfinder Solutions.
9. Erdogmus, Hakan; Morisio, Torchiano. "On the Effectiveness of Test-first Approach to Programming". Proceedings of the IEEE Transactions on Software Engineering, 31(1). January 2005. (NRC 47445). Retrieved 2008-01-14. "We found that test-first students on average wrote more tests and, in turn, students who wrote more tests tended to be more productive."
10. Proffitt, Jacob. "TDD Proven Effective! Or is it?". Retrieved 2008-02-21. "So TDD's relationship to quality is problematic at best. Its relationship to productivity is more interesting. I hope there's a follow-up study because the productivity numbers simply don't add up very well to me. There is an undeniable correlation between productivity and the number of tests, but that correlation is actually stronger in the non-TDD group (which had a single outlier compared to roughly half of the TDD group being outside the 95% band)."
11. Llopis, Noel (20 February 2005). "Stepping Through the Looking Glass: Test-Driven Game Development (Part 1)". Games from Within Retrieved 2007-11-01. "Comparing [TDD] to the non-test-driven development approach, you're replacing all the mental checking and debugger stepping with code that verifies that your program does exactly what you intended it to do."
12. Müller, Matthias M.; Padberg, Frank. "About the Return on Investment of Test-Driven Development" (PDF). Universität Karlsruhe, Germany. p. 6. Retrieved 2012-06-14.
13. Loughran, Steve (November 6, 2006). "Testing" (PDF). HP Laboratories. Retrieved 2009-08-12.

This page is intentionally left blank