Challenges in Teaching Test-Driven Development

Premalatha Sampath
Whitireia New Zealand
450 Queen Street
Auckland campus
Premalatha.Sampath@whitireia.ac.nz

ABSTRACT

Test-driven development (TDD) has been proposed as a solution to improve testing in Industry and in academia. The purpose of this poster is to outline the challenges of teaching a novel Test-First approach in a Level 8 course on Software Testing. Traditionally, introductory programming and software testing courses teach a test-last approach. After the introduction of the Extreme Programming version of AGILE, industry and academia have slowly shifted their focus to the Test-First approach. This poster paper is a pedagogical insight into this shift from the test-last to the test-first approach known as Test Driven Development (TDD).

Keywords: Test driven development, Software testing, Postgrad courses.

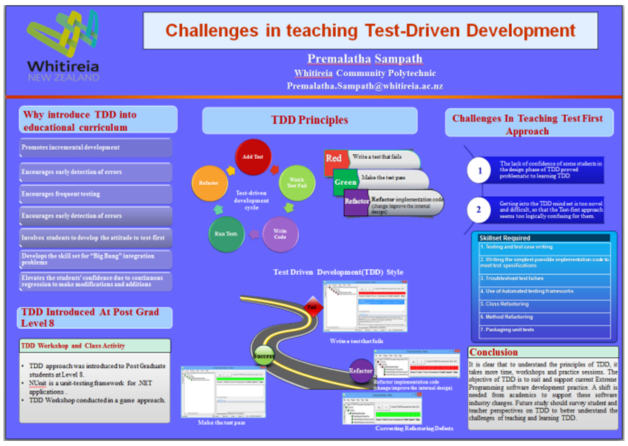
1. INTRODUCTION

TDD, the test driven or test-first design approach to software development has become popular with rise in popularity of agile software development methods, such as Extreme Programming that was introduced around 1988 (Janzen and Saiedian, 2005), but has been practiced since the late 1950’s on projects such as NASA’s Mercury project.

The novel approach that TDD takes to software development is that test-cases are prepared before writing the source code. The test-cases drive system design and development.

In traditional programming, unit cases are constructed after the code is written and then tested by a testing team some time later. In the TDD approach, unit cases are constructed by the programmers before writing source code and testing is performed by the programmers immediately after that unit of source code is written.

TDD takes the test-first approach to an extreme level by always writing test-cases as code, making tests as small as possible and never letting code degrade (Hammond and Umphress, 2012; Janzen et al, 2005). According to Scott Amber (2006), who describes TDD as the simple formula:



Challenges in teaching Test-Driven Development

Whitireia NEW ZEALAND

Premalatha Sampath
Whitireia Community Polytechnic
Premalatha.Sampath@whitireia.ac.nz

Why introduce TDD into educational curriculum

- Promotes incremental development
- Encourages early detection of errors
- Encourages frequent testing
- Encourages early detection of errors
- Involves students to develop the attitude to test-first
- Develops the skill set for “Big Bang” integration problems
- Elevates the students' confidence due to continuous regression to make modifications and additions

TDD Introduced At Post Grad Level 8

TDD Workshop and Class Activity

- TDD approach was introduced to Post Graduate students at Level 8.
- NUnit is a unit-testing framework for .NET applications .
- TDD Workshop conducted in a game approach.

TDD Principles

Red — Write a test that fails
Green — Make the test pass
Refactor — Refactor implementation code (change/improve the internal design)

Test Driven Development(TDD) Style

Fail
Success
Refactor
Write a test that fails
Make the test pass
Refactor implementation code (change/improve the internal design)
Correcting Refactoring Defects

Challenges In Teaching Test First Approach

1. The lack of confidence of some students in the design phase of TDD proved problematic to learning TDD
2. Getting into the TDD mind set is too novel and difficult, so that the Test-first approach seems too logically confusing for them.

Skillset Required
1. Testing and test case writing
2. Writing the simplest possible implementation code to meet test specifications
3. Troubleshoot test failure
4. Use of Automated testing frameworks
5. Class Refactoring
6. Method Refactoring
7. Packaging unit tests

Conclusion

It is clear that to understand the principles of TDD, it takes more time, workshops and practice sessions. The objective of TDD is to suit and support current Extreme Programming software development practice. A shift is needed from academics to support these software industry changes. Future study should survey student and teacher perspectives on TDD to better understand the challenges of teaching and learning TDD.

TDD= Test-first Development (TFD) +Refactoring.

Refactoring, a central aspect of TDD, refers to changing the structure of code without modifying its external behavior. Therefore, it improves the code structure while ensuring it sill passes testing (Madeyski, 2010).

2. WHY INTRODUCE TDD INTO EDUCATIONAL CURRICULUM:

Educators have constantly struggled to improve student’s comprehension and analytic thinking skills. Most of the students rely on a trial and error (Edwards, 2004) approach to fixing errors and debugging (Erdogmus, 2005). In addition, most education curriculum insufficient covers software testing (Keefe, Sheard and Dick, 2006).

TDD’s emphasis on an incremental development, test-first approach and frequent testing has motivated many educators to introduce TDD into education curricula (Desai, Jansen and Savage, 2008; Buffardi and Edwards, 2013). The Virginia Tech Department of Computer Science has been very keen to include TDD in their curricula for the following reasons (Edwards, 2004).

1. It promotes incremental development and encourages always maintaining a working version of the program.
2. It encourages early detection of errors, during designing and writing automated unit tests. Therefore, it promotes frequent testing.
3. It involves students to develop the attitude, not to leave the testing till the end.
4. TDD encourages on incremental development and frequent testing, which develops the skill set for “Big Bang” integration problems.
5. It increases the student’s confidence on their developed coding.
6. It also elevates the students' confidence due to continuous regression to make modifications and additions.
7. The test suites developed in TDD approach serve a progress indicator that allows the student to allows the student to understand how much of the program behavior has been completed.

This poster paper appeared at ITX 2014, incorporating the 5th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2014) and the 27th Annual Conference of the National Advisory Committee on Computing Qualifications, Auckland, New Zealand, October 8-10, 2014. Mike Lopez and Michael Verhaart, (Eds).

Studies and surveys (Desai et al., 2008, Janzen and Saiedian, 2005, Janzen and Saiedian, 2008a, Janzen and Saiedian, 2008b) indicate that the introduction of TDD into education curricula have been met with favorable results both in terms of student's program code quality and software development approach.

3. TDD AT DIFFERENT EXPERIENCE LEVELS

The current pedagogical concern of educators is to determine when to introduce TDD into their curricula (Desai et al., 2008). Various experiments have been conducted to evaluate the level at which it is most appropriate to introduce TDD. Studies tend to show that beginner programmers struggle with TDD as they incorporate frameworks such as Junit or Nunit. Novices who have just started learning what programming is and how it works find it tough enough to understand the purpose of code and hence they find it difficult to understand testing (Spacco and Pugh, 2006). To overcome these testing hurdles, tools to help beginner students and novices have evolved, that give feedback on such things as test coverage and the number of unit tests written and passed (Lance, Sarkar and Bian, 2010), that are meaningful to programming novices.

Several studies have shown that the results of introducing TDD at higher learning levels have been much more promising. Study surveys indicate that mature programmers appreciate the benefits of TDD (Spacco et al., 2006) more than novice programmers. At Whitireia, in Trimester 2 of 2014 the TDD approach was introduced to post graduate students at Level 8.

4. CHALLENGES IN TEACHING TEST FIRST APPROACH

It has been observed that the TDD approach simplifies existing automated testing frameworks, simplifying the conception and implementation of unit test cases, for the implementation of TDD in Nunit with C#. Unit has been introduced. NUnit is an open source unit testing framework for Microsoft .NET. A worksheet is handed to students followed by a 2 day workshop. The worksheet is designed to present the TDD approach in a game called Tic-tac-toe.The requirements represent the game board by creating a 3-by-3 multidimensional array of char data types, with the x-axis representing the horizontal rows and the y-axis representing the vertical columns.

Tests were designed for 3 chosen key features, unit testing was conducted, code refactored in the course of updating code to pass the unit testing. Ultimately, the refactoring of the business code validated the performance of the 3 features.

Firstly, the lack of confidence of some students in the design phase of TDD proved problematic to learning TDD. Secondly, a significant number of students thought that getting into the TDD mind set is too novel and difficult, so that the Test-first approach seems too logically confusing for them.

4.1 Lessons learnt and future plans

It has been observed that TDD as a design approach requires certain skill sets such as:

The future plans are to introduce well-designed in-class activities and homework support learning, to help students develop skills in small steps, and to gradually help them to apply them to more complex problems.

- Testing and test case writing.
- Use of Automated testing frameworks.
- Troubleshoot test failure
- Writing the simplest possible implementation code to meet test specifications.
- Method refactoring
- Class Refactoring
- Packaging unit tests

5. CONCLUSION

The challenges reported here have also been discussed in several other studies (Buffardi et al., 2013, Desai et al, 2008, Janzen et l, 2008a, Janzen et al, 2008b, Madeyski, 2010) that it takes more time, workshops and practice sessions to understand the principles of TDD. The objective of TDD is to suit and support current Extreme Programming software development practice. A shift is needed from academics to support these software industry changes. Future study should survey student and teacher perspectives on TDD to better understand the challenges of teaching and learning TDD.

6. REFERENCES

Ambler, S. W. (2006). Introduction to test driven development (TDD). *Agile Data,[En línea]. Available: http://www. agiledata. org/essays/tdd. html.[Último acceso: 9 June 2012]*.

Buffardi, K., & Edwards, S. H. (2013). Impacts of adaptive feedback on teaching test-driven development*ACM.* Symposium conducted at the meeting of the Proceeding of the 44th ACM technical symposium on Computer science education.

Desai, C., Janzen, D., & Savage, K. (2008). A survey of evidence for test-driven development in academia. *ACM SIGCSE Bulletin, 40*(2), 97-101.

Edwards, S. H. (2004). Using software testing to move students from trial-and-error to reflection-in-action. *ACM SIGCSE Bulletin, 36*(1), 26-30.

Erdogmus, H. (2005). On the effectiveness of test-first approach to programming. IEEE Transactions on Software Engineering, vol. 31, no. 3, pp. 226-237.

Hammond, S., & Umphress, D. (2012). Test driven development: the state of the practice*ACM.* Symposium conducted at the meeting of the Proceedings of the 50th Annual Southeast Regional Conference.

Janzen, D. S., & Saiedian, H. (2005). Test-driven development: Concepts, taxonomy, and future direction. *Computer Science and Software Engineering*, 33.

Janzen, D. S., & Saiedian, H. (2008a). Does test-driven development really improve software design quality? *Software, IEEE, 25*(2), 77-84.

Janzen, D. S., & Saiedian, H. (2008b). Test-driven learning in early programming courses*ACM.* Symposium conducted at the meeting of the ACM SIGCSE Bulletin

Keefe, K., Sheard, J., & Dick, M. (2006). Adopting XP practices for teaching object oriented programming*Australian Computer Society, Inc.* Symposium conducted at the meeting of the Proceedings of the 8th Australasian Conference on Computing Education-Volume 52.

Lance, M., Sarkar, A., & Bian, R. (2010, July 6-9). *Assessing with a unit test framework-variations of approach*. presented at the meeting of the Conference of Computing and Information Technology Research and Education (CITRENZ), Dunedin, New Zealand.

Madeyski, L. (2010). The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment. *Information and Software Technology, 52*(2), 169-184.

Spacco, J., & Pugh, W. (2006). Helping students appreciate test-driven development (TDD)*ACM.* Symposium conducted at the meeting of the Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications.