Alexander Ramsey  
SWEN 603  
July 12, 2016

Test-driven Development Process

An offshoot of the Extreme programming lifecycle, Test-driven development is a process that revolves around the use of a short development cycle combined with requirements that are clearly outlined. These requirements outlines are provided with test cases that either pass or fail, allowing the developer to break down larger problems into smaller sects. In extreme programming, testing occurs more loosely throughout the process. Test driven development provides somewhat more structure in that the testing is driving the development along.

Kent Beck is the author of “Test Driven Development by Example.” This text guides developers through a strict process of unit testing where code is dependent upon test cases that are created. There is a five step sequence in this guide which is repeated throughout the development process in short bursts to ensure that the requirements are met in a fashion that inspires confidence for the developer to continue onto the next issue (Beck, 2003).

The first step in the sequence is to add a unit test. These tests can be written during the creation of user stories and use cases during the requirements phase of development. This first step is what differentiates the test driven development process from others in that the testing is proactive rather than reactive. This allows the developers to think more clearly about the desired functionality before writing any code. It is believed that by following this process, the developer will create something that is more clearly defined for the user in the long run.

The second step in this process is to run the created tests against nothing. They should properly show an expected fail result rather than a pass result. This ensures that the desired outcomes are clearly defined and are in working order before any development begins.

The third step in the process is to develop the program against the test case. At this step, the code is considered to not be final and can be loosely accomplishing the provided test case. Additional debugging may occur at a later step to further hone the compliance of the code to the test case or other functional requirements outside of the scope of the user story.

The fourth step is to run the test cases again, utilizing code in the third step. Following this step, the development process can begin again in the fifth step. The fifth step is the code cleanup and refactoring stage. This stage should serve as a short iteration of development which is more focused on refining the code and readability of the program to improve future adjustments and compatibility with other code.

Below is a graphical representation of this process as a short iterative cycle.

Refactor

**TDD**

Test Code

Draft Code

Design Test

As a process of development, TDD greatly provides benefits in terms of producing accurate and reusable code. Because the developer is focused on meeting requirements, the design is more closely analyzed and used before development begins. This can save time and resources later as additional features are added. In addition, by breaking down larger problems into test cases, there are less errors and issues prevalent. TDD is also considered to be an efficient process that reduces code length and unnecessary development of requirements. By focusing on making the test case pass, the developer is able to save time and not be sidetracked by external issues which may arise.

Relating this process to the ever expanding breadth and depth of modern applications can also highlight some major flaws of TDD. Where requirements are broad in scope and require functional testing across different platforms and development environments, unit tests and TDD as a process may fall short. The process of developing tests in a narrow focus may limit the developer’s ability to think outside of the box, per se. This can lead to tunnel vision, where test cases are designed to pass according to what is believed to be an important aspect of the design of the program. In addition, over testing may become a daunting part of the development process. Thus, slowing the development process and causing a great deal of overhead. This is especially true when the team is too focused on making sure to follow the process, rather than following the flow of problems throughout the development process.

Implementing TDD in a large scale architecture may present some challenges. However, as a mindset it can be very useful. Test driven development is often seen to be something that is inherent in most developers without additional effort, anyways (Latorre, 2014). So, as a process it can be adapted to every development environment. Test driven development is a useful process that allows the problem to be honed and focused on, while worrying about the bigger picture is subsided and managed.

References:

Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003

Latorre, R. (2014). Effects of developer experience on learning and applying unit test-driven development. *IEEE Transactions On Software Engineering*, (4), 381.