Chapter XI

Test–Driven Development:
An Agile Practice to Ensure Quality 
is Built from the Beginning

Scott Mark
Medtronic, Inc.

ABSTRACT

This chapter describes the practice of test-driven development (TDD) and its impact on the overall culture of quality in an organization based on the author’s experience introducing TDD into four existing development projects in an industrial setting. The basic concepts of TDD are explored from an industry practitioner’s perspective before elaborating on the benefits and challenges of adopting TDD within a development organization. The author observed that TDD was well-received by team members, and believes that other teams will have this same experience if they are prepared to evaluate their own experiences and address the challenges.

INTRODUCTION

The spirit of agile methods begins with the promise that whatever software components are built will be of high quality. Agile methods move quality assurance upstream in the software development process, and the most relevant of these methods is the principle of test-driven development (TDD). The essence of TDD is that quality assurance methods are not a sieve through which application code is pushed at the end of a long, drawn out development process. Rather, the development cycle begins with capturing test cases as executable system components themselves. These testing components are then used to drive the development process and deliver components that, by definition, satisfy the quality requirements as they are set out in the test cases. In its purest form, the developer begins by writing a test case that fails and then proceeds to implement the functionality that causes the test to succeed. When this practice is followed carefully, the code that becomes the final product is guaranteed to pass all currently identified test cases.

The goal of this chapter is to explain the changes to the traditional development process in order to drive it with quality assurance, and illustrate the overall impacts on software quality,
process velocity, and developer productivity. The perspectives on TDD presented in this chapter are based on the author’s experience introducing these techniques on four Web application development projects in a large enterprise setting. These projects will be described and the teams’ good and bad experiences with TDD will be explored. The intention of this chapter is to share the experiences, both good and bad, of these teams while using TDD so that other practitioners can anticipate or evaluate similar effects in their own environments.

WHAT IS TEST-DRIVEN DEVELOPMENT (TDD)?

Test-driven development is the practice of implementing executable tests before implementing functional components, and using the activity of testing to propel the implementation of functional components. For purposes of this discussion, the essential components of the test-driven development practice are the following:

- Tests are authored by the developer before implementation.
- Tests are “easily” executed by the developer working on the implementation.
- Tests are at the unit- or component-level.

Tests are Authored by the Developer before Implementation

TDD is a quality improvement process that ultimately is a form of organizational change. A key aspect of this is the transformation of every developer into a tester. Organizations that have separate roles for authoring tests have not completed this transformation—testing will remain a back-and-forth process of transfer. So a requirement of TDD is that the developer who will be implementing functionality begins by writing the test to verify the implementation.

Tests are “Easily” Executed by the Developer Working on the Implementation

This is of course a very subjective metric, but is a key requirement nonetheless. The core of the TDD practice is that running tests is part of the moment-to-moment development process. For TDD purposes, tests should be kept at a practical level of granularity with a consideration toward execution time. Low execution times ensure that these tests can in practice be run frequently and continuously during the work of implementation.

Tests are at the Unit- or Component-Level

There are various levels of testing within the larger landscape of software quality assurance such as unit, functional, system, integration, and user acceptance testing. It is certainly not the goal of TDD to address all of these aspects of testing. TDD promises to increase the amount of quality that is built-in from the start, and encourages developers to think upfront about testability. This is achieved by testing the aspects immediately in front of the developer at the unit- and component-levels of implementation. A unit test is a test that focuses on a given implementation construct (a .java file in Java, a .c file in C/C++, etc.). A component test is a test that focuses on an atomic system component, such as an interface in Java, that might front a number of implementation constructs in the implementation.

As we proceed to higher levels of granularity throughout the system, test-driven development starts to dissolve into more integration-oriented testing methods. As more and more layers are integrated into the working, testable system, the setup and cycle times of execution increases to the point where some of the benefits of test-driven development diminish. Some of these tests can be automated and some can be written before the implementation. Automated tests written
Related Content

Fuzzy Rule-Based Vulnerability Assessment Framework for Web Applications
www.igi-global.com/chapter/fuzzy-rule-based-vulnerability-assessment-framework-for-web-applications/188234?camid=4v1a

Assessing the Usefulness of Testing for Validating and Correcting Security Risk Models Based on Two Industrial Case Studies
www.igi-global.com/article/assessing-the-usefulness-of-testing-for-validating-and-correcting-security-risk-models-based-on-two-industrial-case-studies/136468?camid=4v1a

Context-Aware Privacy and Sharing Control in Collaborative Mobile Applications
www.igi-global.com/chapter/context-aware-privacy-sharing-control/66481?camid=4v1a

Evaluation of Recurrent Neural Network and its Variants for Intrusion Detection System (IDS)
www.igi-global.com/article/evaluation-of-recurrent-neural-network-and-its-variants-for-intrusion-detection-system-ids/204371?camid=4v1a