Quality Assurance of Software Products

Supervised by: Prof. CLAUDIO GIOVANNI DEMARTINI

Author: Ebrahim Kargar Nasrabadi

1 Introduction

The last twenty years has seen major changes in the methodologies, tools and technologies we use to build software, but most significantly, the way we build software has fundamentally been evolving in order to improve the quality, accuracy, and speed of delivering software. By all these regular technological disruptions, growing fast has become essential to survival for software companies.

Emergence of new paradigms in software developments industry not only effected methodologies and the process of development but also tools and technologies has been subject to massive changes in a few decades ago. In this chapter we are going through some of these new paradigm including : Agile Software development, Test-Driven Development , Behavior-Driven Development , Code Refactoring & Review, Pair Programming , Continuous Integration and Continuous Delivery , Continuous Testing and many more.

2 Agile Software Quality Assurance

Software quality is one critical component of the criteria used to measure success of a software development project. Quality Assurance, methodologies and techniques have been accordingly evolved in software industry.

Agile software development methodologies as one of these new paradigms since its inception had improved the quality of the software product in many aspects including: functionality, performance, reusability, efficiency, compatibility…

In this chapter we are examining two agile software quality assurance techniques: Test-Driven Development (TDD) and Behavior-Driven Development (BDD).

TDD is a failing test for a behavior of a piece of code that is written before this behavior is implemented. Test-Driven Development lifecycle includes:

- Write a Statement you wish were true but is not (failing test)
- Add code to make it pass
- Refactor the code

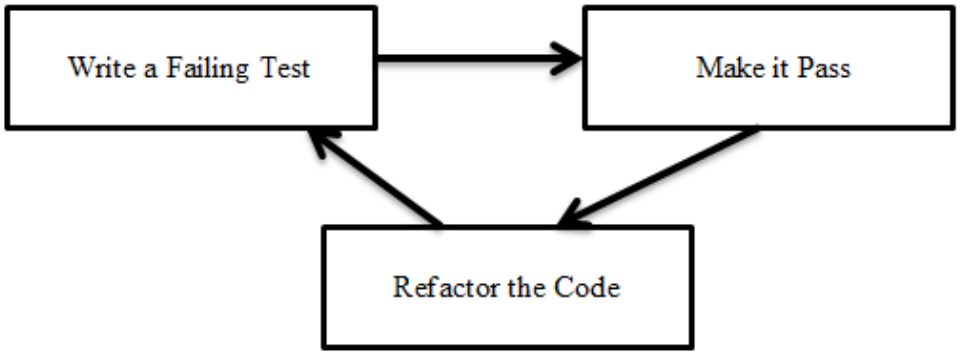

Figure 1: TDD lifecycle

Behavior-Driven Development was originally invented by Dan North in the early to mid-2000s.BDD add a cycle around the TDD cycle, so that you start with a behavior and let that drive your tests, and then let the tests drive the development. Ideally, BDD is driven by some kind of acceptance test, but that's not 100% necessary.

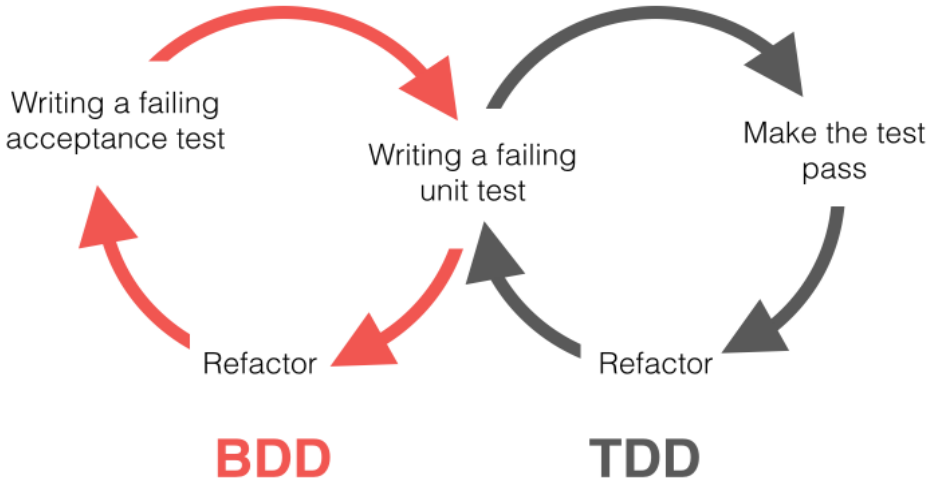

Figure 2: BDD lifecycle

3 Performance Testing

Performance testing is measuring how a software product would behaves under an increasing load (both numbers of users and data volumes).Load testing is verifying that the system can operate at the required response times when subjected to its expected load. Stress testing is finding the failure point(s) in the system when the tested load exceeds that which it can support.

In order to accurately measure performance of a software product, we need to take into account certain key performance indicators (KPIs), some of the most important software performance metrics are:

- Availability: complete inability of an end user to make effective use of the application either because the application is simply not responding or response time has degraded to an unacceptable degree.

- Response time: the time between the end user requesting a response from the application and a complete reply arriving at the user's workstation.

- Throughput: the rate at which application-oriented events occur. A good example would be the number of hits on a web page within a given period of time.

- Utilization: The percentage of the theoretical capacity of a resource that is being used. Examples: how much network bandwidth is being consumed by application traffic or the amount of memory used on a web server farm when 1,000 visitors are active.

In this chapter we are going through introduction and of some the novel tools used to measure performance of a software product, in addition we would practice some of them to measure above-mentioned quality metrics of the software.

4 Automation Testing

Software quality assurance is an integral, costly, and time-consuming activity in software development lifecycle. Although agile doctrine presents agenda of continuous quality improvement on the software, it is usually limited to current iteration. Agile practices include very little quality practices at the release of artifacts, which are considered essential for a quality software product. In addition, because testing involves running the system being tested under a variety of configurations and circumstances, automation of execution-related activities offers another potential source of saving efforts in the quality assurance process.

A test automation framework is an integrated system that sets the rules of automation of a specific product which integrates several components such as function libraries, test data sources, object details and various reusable modules. These components help building a suitable automation framework which enable testing the business process as per the requirements. In this chapter we will dive into some of the testing automation frameworks and we will discuss how we could employ automation on a resource-intensive test suite used by an actual testing organization.

5 Appendix & Future Work

As a matter of putting some of these stat-of-art quality software assurance techniques in practices, in appendix I attached result of my experiments in a telecom company by working on the quality of Microservices as a product.

Although there are some good practices in proposing a comprehensive software quality assurances framework ,to take full gain and potential of agile methodologies, introducing an all-inclusive quality assurances framework that could enables measurement of all the existed quality metrics in several software product is missed in software industry.