Chapter 13

Behavior-Driven Development Using Specification by Example: An Approach for Delivering the Right Software Built in Right Way

Praveen Ramachandra Menon
Independent Researcher, Singapore

ABSTRACT

This chapter highlights a crucial problem seen often in software development that is bridging the communication gap between business and technical language and that it can be addressed with “Behavior Driven Development” (BDD) methodology supplemented with “Specification By Example” approach of delivering the right software that matters. Effective communication has always been a challenge between clients, business stakeholders, project managers, developers, testers and business analysts because a “ubiquitous” language that everyone can easily understand and use does not exist. Specification By Example serves as that ubiquitous language for all, helps build right software that matters through effective communication. Specifications are written in plain English language using the Gherkin syntax to describe various behaviors of software. BDD tools help write software specification using gherkin language and also create a living documentation that is automatically generated by programming language reflecting the current state of software at any given point of time.

INTRODUCTION

Behavior Driven Development (BDD) is an emerging practice in agile software development (North, 2006). BDD combines the general techniques and principles of Test Driven Development (TDD), Acceptance Test Driven Development (ATDD), Domain-Driven Design and object-oriented design. It provides software development and management teams an easy way to collaborate and create a shared understanding of system by building a domain vocabulary in English language that all the team members from various backgrounds can easily understand and use for communication with each other.

DOI: 10.4018/978-1-4666-9858-1.ch013
This chapter will highlight the key concepts involved from inception to completion, while building software using BDD technique in conjunction with specification by examples approach. The chapter will take the readers step by step through a real time example of building the vending machine software implemented in Java. The chapter presents an immense opportunity to learn about agile practices involving BDD and TDD, which are some of the latest emerging trends. The whole BDD life cycle that also includes TDD implementation internally would be explained in detail using various diagrams and code snippets. This chapter starts with some background on history and evolution of the software development from early 1990s, definition and simplification of concept, author’s hypothesis on BDD implementation, some controversies, myths, benefits and key challenges involved with BDD implementation and concludes with a real time example of building the software in a BDD way.

BDD when used in conjunction with specification by example approach helps reflect on some of the lean principles in software development by avoiding wasteful over-specification. BDD avoids spending time on details of requirements that keep on changing before even being developed. BDD gives an efficient way to perform end user perspective regression checks on the system and validates frequently whether actual behavior is as per specification with help of automated continuous code integration, build and deployment process. BDD provides most recent (almost instant) and reliable living documentation with minimal maintenance costs that truly reflects the current state of software being developed. When the specification is described with concrete examples, it becomes very easy to develop and test system and uncover ambiguities in behavior. Once this specification is automated it becomes an executable acceptance test. BDD practice can be easily fit into either short agile iterations or flow-based process, so that information on upcoming work is produced just in time (Adzic, 2009).

In BDD, ideally everyone in the team discusses and writes the specification with concrete examples. A team typically has developers, testers, business analysts etc. In agile teams at the least, product owner write the specification so that team will have clarity on what they need to develop. In BDD the acceptance tests focus on various behavior of system and will initially fail, as features are not yet implemented. As a sprint progress, the developers will implement the features just enough to make test cases merely pass. Different kinds of test code such as unit, integration and system or user acceptance tests are plugged in to automation framework that will internally pull down the inputs from specification. It validates expected output data from spec against the actual output obtained from system under test. Once the test cases pass, feature is marked complete. Later all the code are optimized and refactored for maintainability, efficiency and reusability.

This approach ensures higher product quality, setting clear expectations for all and the validation process becomes more efficient. This approach also leads to less rework as team collaboratively ensure a shared common understanding among all and thereby allowing better alignment of activities among different kinds of roles on a given project resulting into a flawless delivery.

BACKGROUND

In order to have a clear understanding on today’s context of software development, it would be helpful to glance through the history and evolution of software development over past decades. From time to time the software development community has attempted to solve the classic problems emerged in industry.
Related Content

Service-Oriented Agents and Meta-Model Driven Implementation
www.igi-global.com/chapter/service-oriented-agents-meta-model/28959?camid=4v1a

Bridging Services and Resources with Structural Services
www.igi-global.com/article/bridging-services-and-resources-with-structural-services/178565?camid=4v1a

A Petri Net-Based Tool for the Analysis of Generalized Continuous Time Bayesian Networks
Daniele Codetta-Raiteri and Luigi Portinale (2014). Theory and Application of Multi-Formalism Modeling (pp. 118-143).
www.igi-global.com/chapter/a-petri-net-based-tool-for-the-analysis-of-generalized-continuous-time-bayesian-networks/91945?camid=4v1a

Information Theoretic XSS Attack Detection in Web Applications
www.igi-global.com/article/information-theoretic-xss-attack-detection-in-web-applications/118145?camid=4v1a