Programowanie sterowane zachowianiami - BDD

Mateusz Miotk

23 kwiecień 2018

Instytut Informatyki UG

Współpraca w ramach całego zespołu

Dotychczas wszystkie poznane techniki stosowano głównie do programistów. Pomijano tutaj klientów, analityków i innych członków zespołu, którzy w głównej mierze nie potrafią zrozumieć kodu. Ponadto pisana dotychczas dokumnetacja w głównej mierze służy programistom a nie innym członkom zespołu. Dlatego wprowadza się pojęcie programowania sterowanego zachowaniami - **BDD** (Behaviour Driven Development).

Co to jest BDD

BDD to proces zwinny, zaprojektowany po to, by w trakcie całego projektu zachować koncentrację na generowaniu wartości dla interscenariuszy. BDD jest odmianą TDD. Najpierw definiuje się specyfikacje, a potem na ich podstawie pisze się kod rozwiązania. W odróżnieniu od TDD, gdzie używane są testy jednostkowe, w BDD przed rozpoczęciem pisania kodu tworzone są liczne specyfikacje (nazywane scenariuszami). W BDD zwykle tworzone są wymagania funkcjonalne z wyższego poziomu. Kolejną różnicą są odbiorcy. BDD ma umożliwiać udział w pracach wszystkim (programistom, testerom, menadżerom, użytkownikom, klientom itd.). BDD funkcjonuje od "zewnątrz do wewnątrz" (najpierw opisywane są funkcje, a potem zespół tworzy jednostki).

BDD i TDD

Scenariusze należy zacząć od zdefiniowania funkcji, zachowań, opracować je, stosując TDD razem z testami jednostkowymi, a po ukończeniu prac należy sprawdzić poprawność rozwiązania za pomocą BDD. Zakończenie jednego scenariusza BDD może zająć kilka godzin, a nawet kilka dni.

Narracja w BDD

Historia w BDD składa się z jednej narracji powiązanej z przynajmniej jednym scenariuszem. Narracja pełni funkcję czysto informacyjną, a jej głównym zadaniem jest zapewnienie danych umożliwiających komunikację między wszystkimi zainteresowanymi stronami. Narracja to krótki opis funkcji przedstawiony z perspektywy osoby, która tej funkcji potrzebuje.

Aby napisać narracje należy odpowiedzieć na trzy podstawowe pytania:

- Jaką wartość ma dawać funkcja, którą należy zbudować?
- Kto potrzebuje żądanej funkcji?
- Czym jest funkcja, którą należy opracować lub jaki jest cel prac?

Przykład narracji

In order to: Aby wydajne było wyszukiwanie przyjaciół
As a: Jako administrator forum
I want to: Chce móc posortować leksykograficznie przyjaciół

Scenariusze

Kiedy narracja ma umożliwiać komunikację, to scenariusze mają być efektem tej komunikacji. Powinny opisywać interakcje osoby pełniącej daną funkcję (wskazanej w punkcie **Jako**) z systemem. Inaczej niż testy jednostkowe, pisane w formie kodu przez programistów dla programistów, scenariusze w BDD należy definiować za pomocą zwykłego języka i podawać w nich minimalną ilość szczegółów technicznych, tak by wszystkie osoby zaangażowane w projekt miały wspólny opis zachowań (funkcji), które zostaną dodane do systemu.

Opis scenariuszy

Każdy scenariusz składa się z opisu oraz jednego bądź większej liczby kroków rozpoczynających się od słów (zakładając, że; jeśli; to). Opis jest krótki i pełni funkcję informacyjną. Pomaga szybko zrozumieć czego scenariusz dotyczy. Natomiast kroki to sekwencja warunków wstępnych, zdarzeń i oczekiwanych skutków przebiegu scenariusza. Kroki pomagają jednoznacznie zdefiniować zachowanie i można je łatwo przekształcić w automatyczne testy.

Przykład scenariusza

```
Narrative:
In order to: Aby wydajnie zarządzać ofertą księgarni
As a: Jako administrator sklepu
I want to: Chcę móc dodawać, aktualizować i usuwać książki

Scenario: Użytkownik powinien móc wyświetlić informacje
o książce

Given: Zakładając, że użytkownik znajduje się na stronie
z książkami
When: Jeśli użytkownik wybierze książkę
Then: To formularz będzie zawierał wszystkie dane
```

Opis kroków w scenariuszu

Krok “zakładając, że” (Given) definiuje kontekst lub warunki wstępne, które muszą być spełnione, by dalsza część scenariusza mogła zakończyć się sukcesem.

Krok “jeśli” (When) definiuje działania lub zdarzenia.

Krok (Then) określa zakończenie scenariusza. Każdy scenariusz powinien kończyć się tą klauzulą.

Programowanie sterowane zachowaniami - JBehave

JBehave (http://jbehave.org/) to platforma do stosowania BDD dla aplikacji napisanych w Javie. JBehave służy do pisania testów akceptacyjnych, które można wykonywać i zautomatyzować. Kroki z historii są związane z kodem w Javie za pomocą zestawu udostępnianych przez platformę adnotacji.

Integracja JBehave z IDE:

- Eclipse (http://jbehave.org/eclipse-integration.html)
- IntelliJ (https://plugins.jetbrains.com/category/23-jbehave/idea)
- NetBeans (http://plugins.netbeans.org/plugin/50532/jbehave-navigator-plugin)

Tworzenie aplikacji JBehave w Maven (kompilacja: `mvn verify`)

`mvn archetype:generate -Dfilter=org.jbehave:jbehave` (opcja 5)

Bibliografia

- V. Farcic, A. Garcia, *TDD. Programowanie w Javie. Sterowanie testami, Wydawnictwo Helion, 2016.*
- J. F. Smart *BDD w działaniu. Sterowanie zachowaniem w rozwoju aplikacji, Wydawnictwo Helion, 2016.*
- http://jbehave.org/