УДК 004.054

Ткачик Д.А.

РОЗРОБКА ЧЕРЕЗ ПОВЕДІНКУ, ЯК ОДНА З МЕТОДОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

[1] Вінницький національний технічний університет;

Анотація

Описано одну з методологій розробки програмного забезпечення, обгрунтовано плюси та мінуси даного підходу, а також представлено "Karate" як інструмент описаної методології .

Ключові слова: тестування, розробка, програмне забезпечення, програмування, розробка через поведінку.

Abstract

One of the software development methodologies is described, the pros and cons of this approach are substantiated, and the "Karate" is presented as a tool of the described methodology

Keywords: testing, development, software, programming, development through behavior.

Вступ

Сучасний процес розробки програмного забезпечення потребує не лише якісного мануального тестового покриття відносно готового продукту, а й покриття автоматизованими тестами на ранніх етапах розробки. І сьогодні це не лише ознака гарного тону в процесі розробки, а важлива вимога до написання коду вцілому. Проте розробка автоматизованих тестів потребує наявності як чіткої та детальної документації, так і ефективної взаємодії між програмістами та інженерами по забезпеченню якості. Використовуючи методологію розробки через тестування (TDD), найчастіше код автоматизованих тестів пише сам програміст, який, власне, відповідає за розробку того чи іншого функціоналу, причому тести покривають саме програмний код, а не систему з тієї точки зору, з якої її бачить користувач. Більше того, розроблені автотести будуть мало зрозумілі для інших учасників життєвого циклу програмного забезпечення, окрім як програмістів. Саме тому в даній статті пропонується більш зручний підхід до впровадження автоматизованого тестування в процес розробки програмного забезпечення.

Загальний опис BDD методології

Behavior Driven Development (BDD – розробка через поведінку) - це процес розробки програмного забезпечення, який виник з Test Driven Development (TDD). BDD підхід створений для того, щоб виправити проблеми, які можуть виникнути при використанні TDD, а саме, забезпечити краще взаєморозуміння всередині команди, тобто не тільки для розробників, полегшити підтримку коду через наочне уявлення про його функціональність, тести і їх результати виглядають більш "людяно", полегшується процес міграції при переході на іншу мову програмування.

За словами Дена Норта, відповідального за розвиток BDD, «BDD використовує приклади на різних рівнях для створення загального розуміння і поверхневої невизначеності для надання програмного забезпечення, яке має значення».

Хоча принциповою ідеєю BDD є те, що розробка програмного забезпечення має керуватися як бізнес-інтересами, так і технічним розумінням, практика BDD дійсно передбачає використання спеціалізованих програмних засобів для підтримки процесу розвитку.

BDD в значній мірі полегшується завдяки використанню простої предметно-орієнтованої мови (DSL) з використанням природних конструкцій, які можуть описати поведінку і очікувані результати. Тестові сценарії вже давно є популярним застосуванням DSL, з різним ступенем складності. BDD вважається ефективною технічною практикою, особливо коли «проблему простору» бізнес-завдання вирішити складно.

Працюючи за методологією BDD, автотестування і складання специфікації супроводжує кожен етап циклу розробки ПЗ, що забезпечує постійну актуальність автоматизованих тестів та відповідної документації.

Тож перш за все, необхідно обрати інструмент, який допоможе швидше писати автоматизовані тести і буде простим у обслуговуванні.

Наявність правильних процесів, інструментів і технічних рішень для автоматичних тестувань API стає важливим, як ніколи раніше. І за допомогою тенденції shift-left, тестування API стає більшим, ніж просто рішення по контролю якості, тепер це критично важливий компонент успішної безперервної інтеграції та розгортання програмного забезпечення.

Karate як інструмент BDD тестування

Karate - це інструмент автоматизації тестування API з відкритим кодом. Тести API записуються за допомогою синтаксису Gherkin. На відміну від інших фреймворків BDD, таких як Cucumber, Specflow або JBehave, Karate допомагає створювати сценарії BDD-тестів без необхідності написання характеристик етапів. Необхідні характеристики генеруються самим Karate DSL, що прискорює і спрощує процес запуску тестування.

Визначимо переваги такого рішення:

1. інтегрований на основі HttpClient та Cucumber-JVM;
2. може запускати тестування і отримувати звіти, як інші стандартні інструменти Java;
3. підтримує паралельне багатопотокове виконання програмного коду, а також конфігурацію перемикання / просування;
4. створити тест можна без знань мови програмування Java;
5. тестування може здійснювати людина, яка не працюючий безпосередньо в програмуванні.

У Karate є багато особливостей, якими варто скористатися, а саме:

1. імітація сервлетів;
2. графічний інтерфейс для візуального виправлення тестових сценаріїв;
3. вивчення файлів різних видів для створення змінних;
4. використання feature-файлів зі змісту іншого feature-файлу;
5. використання (виклик) програмного коду Java;
6. застосування тестового сценарію, наприклад тестів продуктивності Gatting.

Так як продукт створений на основі мови Java, його можна запускати практично звідки завгодно, особливих технічних і системних обмежень розробниками не встановлено. Новий проект створюється за допомогою різноманітних шаблонів, які в достатній кількості присутні в мережі. Наприклад, можна використовувати персональні шаблони Maven.

Karate надає повністю автономний JAR, що вимагає запуску файлів без певних залежностей або конфігурації. Система бере потрібні файли, вибудовує коректні шляхи до них і виконує їх, створюючи вичерпні Cucumber-звіти.

Висновок

Дана методологія дозволяє покращити тестування програмного забезпечення та забезпечити якість продукту, а також дозволяє в більш зрозумілій формі проводити аналіз розроблених систем. Завдяки такому інструменту як Karate, розробники можуть швидко та ефективно реалізовувати автоматизовані тести за допомогою мови програмування Java.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Behavior-driven development [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Behavior-driven_development — назва з екрану.

2. Haring, R. Behavior Driven development: Beter dan Test Driven Development/R. Harring. — Veen Magazines, 2011.

3. John F. BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle/ F. John. — 2014. — 384 с. — ISBN 978-1617291654.

.

Денис Анатолійович Ткачик **–** аспірант кафедри АІІТ, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, м. Вінниця, e-mail: true.tkachyk@gmail.com

Denys A. Tkachyk – AIIT graduate student, Department of Computer Systems and Automation, Vinnytsia national technical University, Vinnytsia, e-mail: true.tkachyk@gmail.com