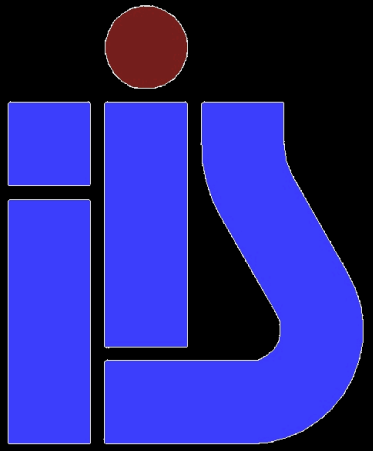


Automatyzacja kryteriów akceptacji

(T -> B)DD

Testowanie akceptacyjne

- Testowanie formalne przeprowadzane w celu umożliwienia użytkownikowi, klientowi lub innemu uprawnionemu podmiotowi ustalenia, czy zaakceptować system lub modułu.

- Źródłem informacji do projektowania kryteriów (testów) akceptacyjnych jest specyfikacja wymagań
  - Specyfikacja steruje wytwarzaniem (od ogółu do szczegółu)

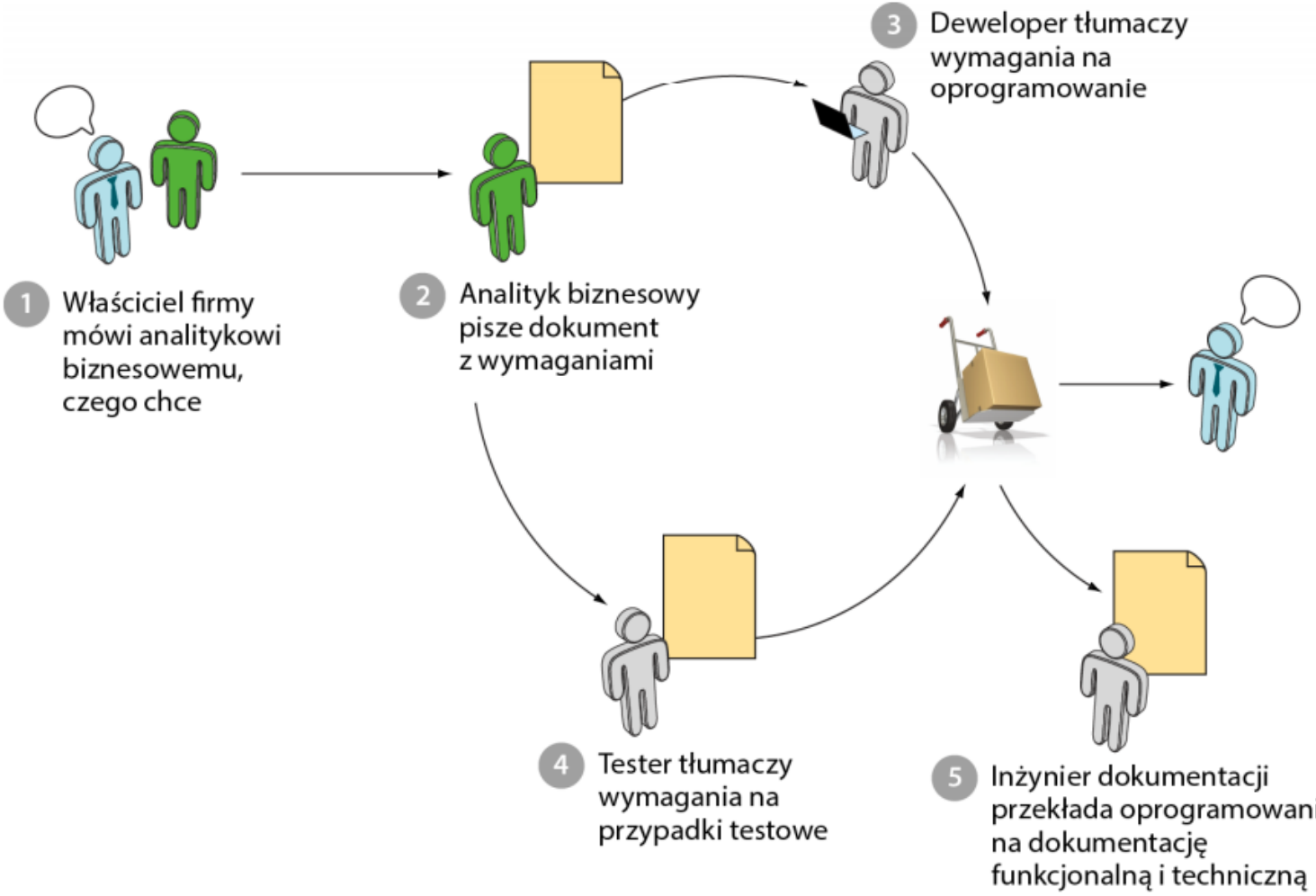
Specyfikacje systemu

- W dużej mierze opisują zachowanie systemu
  - Definiowane przez analityków w trakcie analizy wymagań
- Wymagają języka pozwalającego na komunikację pomiędzy członkami zespołu projektowego a biznesem (interesariuszami – ang. stakeholders)
  - Język naturalny, zrozumiały dla biznesu
  - Język uniwersalny umożliwiający jednoznaczne definiowanie wymagań

Cele biznesowe i cechy funkcjonalne

- Koncentracja na cechach funkcjonalnych, które dostarczają wartości biznesowych
  - Cecha funkcjonalna – namacalny, dający się zrealizować fragment funkcjonalności, który pomaga w osiągnięciu określonego celu biznesowego
- Np. (system bankowy):
  - Cel biznesowy:
    - Przyciągniecie większej liczby klientów dzięki zapewnieniu im wygodnego i prostego sposobu zarządzania kontami
  - Cechy funkcjonalne (budowane stopniowo)
    - Przelew środków pomiędzy własnymi rachunkami klienta
    - Przelew środków na inny rachunek w kraju
    - Przelew środków na rachunek za granicą

Dodawanie nowej cechy do systemu - klasycznie

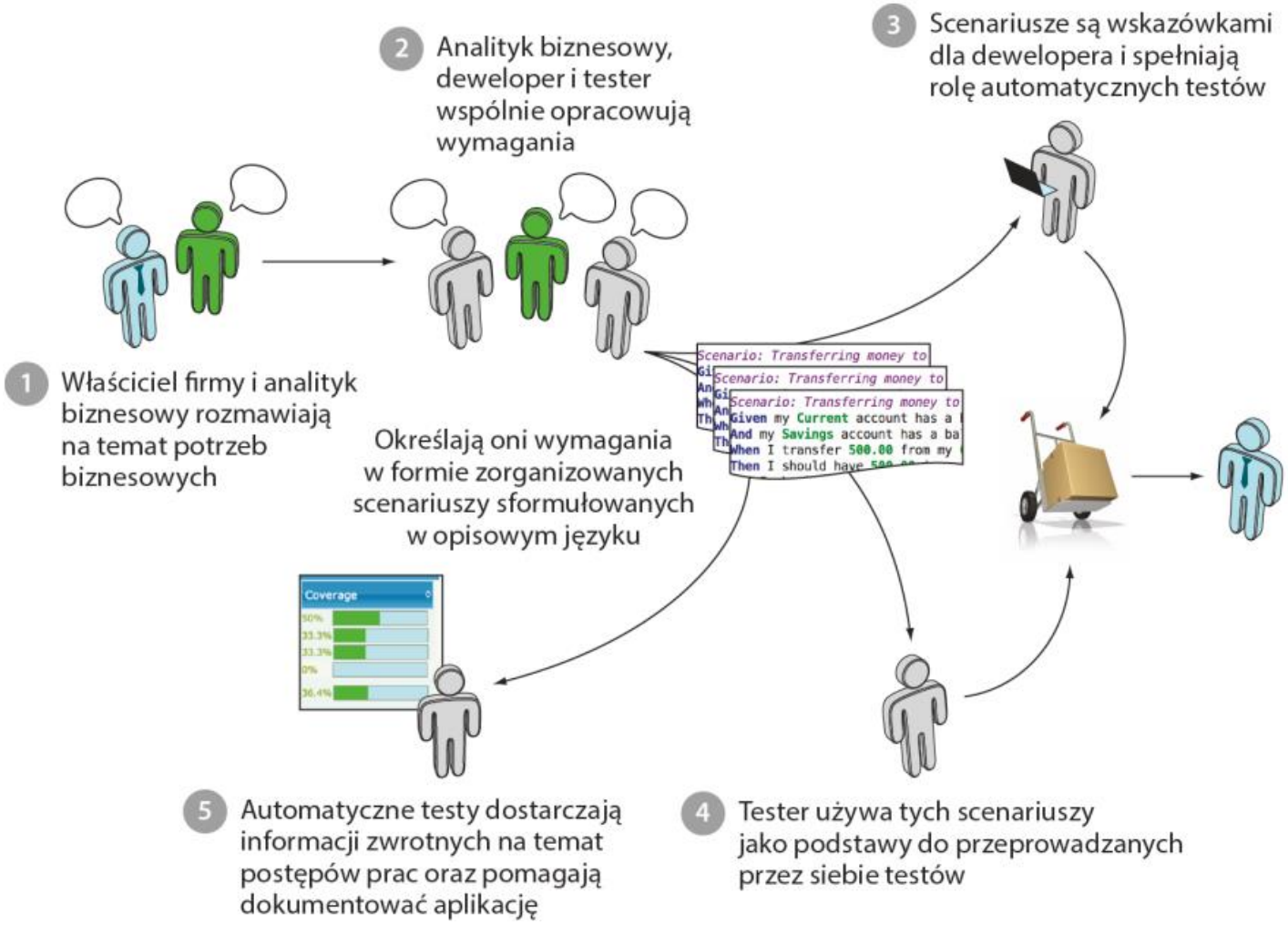


Źródło: John Ferguson Smart, BDD w działaniu, Helion 2016

Behavior Driven Development

- BDD
  - Zbiór praktyk inżynierii oprogramowania, których celem jest ułatwienie szybszego dostarczania wartościowego oprogramowania cechującego się dobrą jakością
  - Bazuje na praktykach zwinnych
  - Pierwotnie zaprojektowana jako „lepsze TDD”

Dodawanie nowej cechy do systemu - BDD

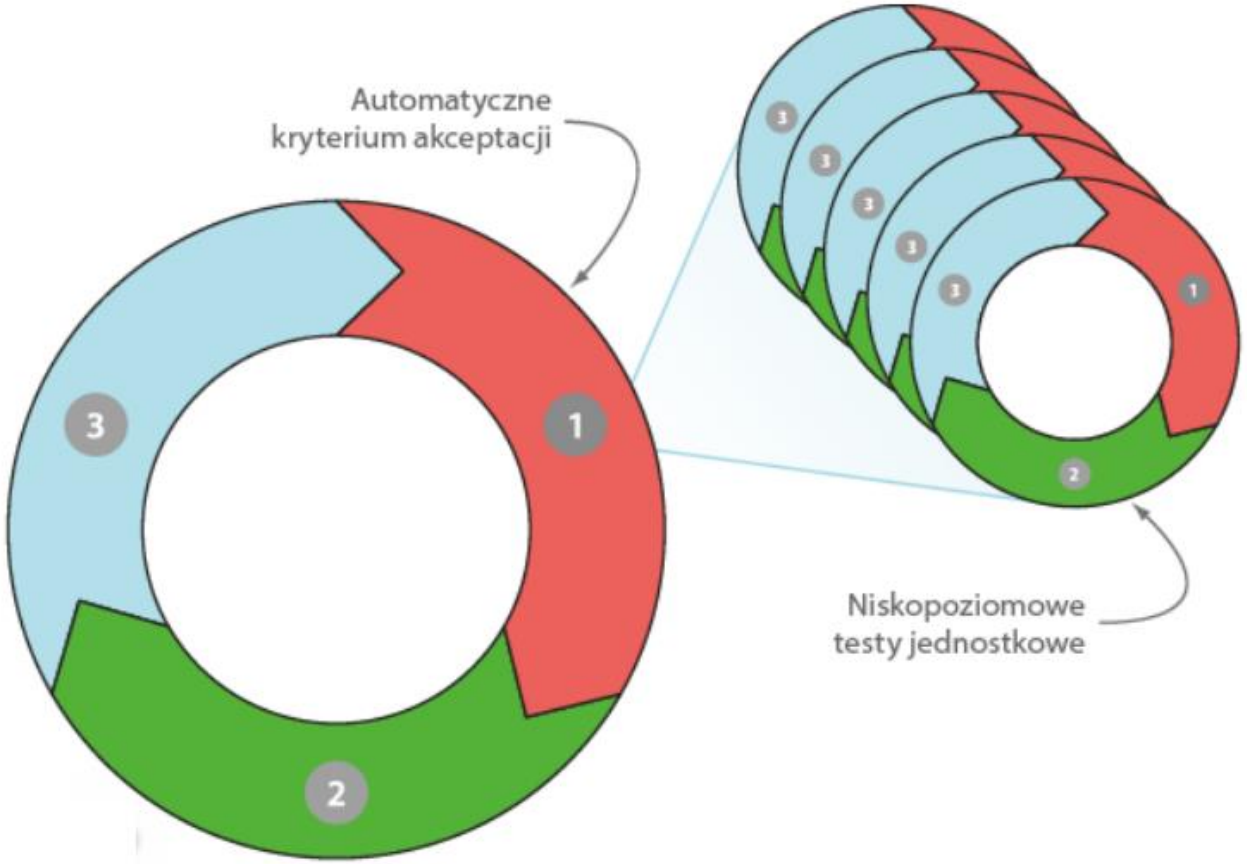


Źródło: John Ferguson Smart, BDD w działaniu, Helion 2016

Automatyczne testy akceptacyjne

- Wyraźna komunikacja zamiaru i dostarczanie sensownych informacji nt. bieżącego stanu aplikacji
- Bardzo ważna jest umiejętność pisania niezawodnych testów akceptacyjnych
  - Odporność na upływ czasu – entropia
  - Koszty utrzymania przy wzroście bazy kodu

BDD a TDD


Automatyczne
kryterium akceptacji
3
1
2
Niskopoziomowe
testy jednostkowe

[B|T]DD jako dokumentacja

- TDD - Dokumentacja techniczna – dla specjalistów
  - Uwzględnianie przypadków granicznych, wyjątkowych

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class MyTests {

  @Test
  public void multiplicationOfZeroIntegersShouldReturnZero() {

    // MyClass is tested
    MyClass tester = new MyClass();

    // assert statements
    assertEquals("10 x 0 must be 0", 0, tester.multiply(10, 0));
    assertEquals("0 x 10 must be 0", 0, tester.multiply(0, 10));
    assertEquals("0 x 0 must be 0", 0, tester.multiply(0, 0));
  }

}
```

- Dokumentacja wysokopoziomowa – zrozumiała dla interesariuszy (ang. stakeholders)
  - Kryteria akceptacji niekoniecznie pokrywają przypadki graniczne

```
Feature: GoogleSearch
  I want to find some cheese

Scenario: Finding some cheese
  Given I am on the Google search page
  When I search for "Cheese"
  Then the page title should start with "cheese"
```

Testowanie systemu

- Od wewnątrz do zewnątrz (Inside-out/ bottom-up, classic school)
  - Zaczynamy od poziomu komponentu
- Od zewnątrz do wewnątrz – (Outside-in top-down, London school)
  - Zaczynamy od wysokopoziomowych kryteriów akceptacji

Poznajmy język BDD - Gherkin

- Zrozumiały dla biznesu język domenowy
  - Opis zachowania systemu bez szczegółów opisujących sposób jego implementacji
  - Umożliwia specyfikowanie oraz automatyzacje testów
    - Wykonywalne specyfikacje
  - Gramatyka języka występuje w kilkudziesięciu wersjach językowych

Gherkin - składnia

- Zorientowany liniowo (wcięcia definiują strukturę)
- Podstawowa struktura
  - Funkcja/cecha (Feature)
    - Scenariusz (Scenario)
      - Krok (Step)
- Konwencje
  - Jedna funkcja na plik
  - Plik .feature

```
Feature: Some terse yet descriptive text of what is desired
  Textual description of the business value of this feature
  Business rules that govern the scope of the feature
  Any additional information that will make the feature easier to understand

  Scenario: Some determinable business situation
    Given some precondition
      And some other precondition
     When some action by the actor
      And some other action
      And yet another action
     Then some testable outcome is achieved
      And something else we can check happens too

  Scenario: A different situation
      ...
```

Kroki scenariusza

- Given – When – Then
  - Given – zdefiniowanie stanu początkowego systemu
    - np. Utworzenie rekordów, zalogowanie użytkownika
  - When - opisanie kluczowych akcji wykonywanych przez użytkownika
    - Interakcja z interfejsem (użytkownika)
  - Then
    - Obserwacja efektów
  - + And, But

Szablon scenariusza i tabele przykładów

- Pozwalają na unikanie zdublowanych tekstów

```
Scenario: eat 5 out of 12
  Given there are 12 cucumbers
  When I eat 5 cucumbers
  Then I should have 7 cucumbers

Scenario: eat 5 out of 20
  Given there are 20 cucumbers
  When I eat 5 cucumbers
  Then I should have 15 cucumbers
```

```
Scenario Outline: eating
  Given there are <start> cucumbers
  When I eat <eat> cucumbers
  Then I should have <left> cucumbers

  Examples:
    | start | eat | left |
    |  12   |  5  |  7   |
    |  20   |  5  |  15  |
```

Założenia

- Uruchamiane przed każdym scenariuszem
  - Pozwalają na uniknięcie powtarzania tych samych kroków w scenariuszach

```
Feature: Multiple site support
  As a Mephisto site owner
  I want to host blogs for different people
  In order to make gigantic piles of money

Background:
  Given a global administrator named "Greg"
  And a blog named "Greg's anti-tax rants"
  And a customer named "Dr. Bill"
  And a blog named "Expensive Therapy" owned by "Dr. Bill"

Scenario: Dr. Bill posts to his own blog
  Given I am logged in as Dr. Bill
  When I try to post to "Expensive Therapy"
  Then I should see "Your article was published."

Scenario: Dr. Bill tries to post to somebody else's blog, and fails
  Given I am logged in as Dr. Bill
  When I try to post to "Greg's anti-tax rants"
  Then I should see "Hey! That's not your blog!"

Scenario: Greg posts to a client's blog
  Given I am logged in as Greg
  When I try to post to "Expensive Therapy"
  Then I should see "Your article was published."
```

Grupowanie scenariuszy (tagi)

- @nazwa
  - Pozwalaja grupować scenariusze (i potencjalnie wykonywać) w zależności od określenia docelowego znacznika
    - Np. @iteration-1, @end-to-end

```
@end-to-end
Scenariusz: Rejestrowanie online nowego konta uczestnika programu Frequent Flyer
...
```

Wskazówki budowania scenariuszy

- Scenariusze powinny być:
  - Deklaratywne (co a nie jak)
  - Pozbawione informacji o typie interfejsu (WebUI, GUI, konsola, Voice, Mobile)

- Declarative programming
  - Saying *what* you want
- Procedural programming
  - Saying *how* to achieve it

Declarative
A tower of 3 blocks.

Procedural
1. Put down block A.
2. Put block B on block A.
3. Put block C on block B.

Narzędzia BDD

- **Cucumber**
  - Opracowany dla języka Ruby, obecnie wspiera wiele platform
  - Oparty na języku Gherkin

	Ruby/JRuby
	JRuby (using Cucumber-JVM)
Java	Java
Groovy	Groovy
JS	JavaScript
JS	JavaScript (using Cucumber-JVM and Rhino)
	Clojure
.Gs GOSU	Gosu
Lua	Lua
Microsoft .NET	.NET (using SpecFlow)
php	PHP (using Behat)
python	Jython
	C++
	Tcl

JBehave

- Framework BDD dla języka Java
  - posiada swoją składnie dla wykonywalnych specyfikacji podobną do Gherkin (istnieje możliwość wykorzystania bezpośredniego języka Gherkin)

```
Narrative:
In order to develop an application that requires a stack efficiently
As a development team
I would like to use an interface and implementation in Java directly


Scenario:  Basic functionality of a Stack

Given an empty stack
When the string Java is added
And the string C++ is added
And the last element is removed again
Then the resulting element should be Java

Scenario:  Stack search

Given an empty stack
When the string Java is added
And the string C++ is added
And the string PHP is added
And the element Java is searched for
Then the position returned should be 3
```

Automatyzacja scenariuszy w cucumber-jvm

```
Scenariusz: Przelew środków na rachunek oszczędnościowy
  Zakładając, że mam rachunek "bieżący" z saldem 1000.00 PLN
  I mam rachunek "oszczędnościowy" z saldem 2000.00 PLN
  Gdy przeleję 500.00 PLN z rachunku "bieżący" na rachunek "oszczędnościowy"
  Wtedy powinienem miec saldo 500.00 PLN na moim rachunku "bieżący"
  I powinienem miec saldo 2500.00 PLN na moim rachunku "oszczędnościowy"
```

```
@Zakładając("^mam rachunek \"(.*)\" z saldem (\\d+\\.\\d+) PLN$")
public void setupAccount(String accountType, double amount){
    //...
}
```

```
@Gdy("^przeleję (\\d+\\.\\d+) PLN z rachunku \"(.*)\" na rachunek \"(.*)\"$")
public void transfer(double amount, String sourceAccount, String destAccount){
    //...
}
```

```
@Wtedy("^powinienem miec saldo (\\d+\\.\\d+) PLN na moim rachunku \"(.*)\"$")
public void balance(double balance, String account){
    //...
}
```

Selektywne wykonywanie kroków

```
@end-to-end
Scenariusz: Rejestrowanie online nowego konta uczestnika programu Frequent Flyer
...
```

To jest adnotacja Cucumber a nie JUnit

```
@Before("@end-to-end")
public void initializeDatabase() {
    TestDatabase.initialize();
}
```