

Wprowadzenie do testowania oprogramowania



Behaviour Driven Development Podstawy

Inspiracja

- James Bach? Robert C. Martin? ... ?
- Marty Cagan from [svpg](#)?
- Desingers? [Product Mgmt](#)?
- [Startups](#)?
- Pragmatyczny programista? Design Patterns?

Ostatnie zajęcia

- Wasze pytania
- Przypadek testowy

Ćwiczenie

1. Problem
2. Mierzymy się z zadaniem samemu [timebox]
3. Pytamy
4. Zrobiłam / Zrobiłem - warto pokazać
5. Chyba, że wykładowca nie poprosi, copy&paste zakazane

Rekomendacja: własny laptopem*

Behaviour Driven Development

BDD

- Jak się dogadać?



www.projectcartoon.com

What the business people originally had in mind



www.projectcartoon.com

How the product was described in the specifications document

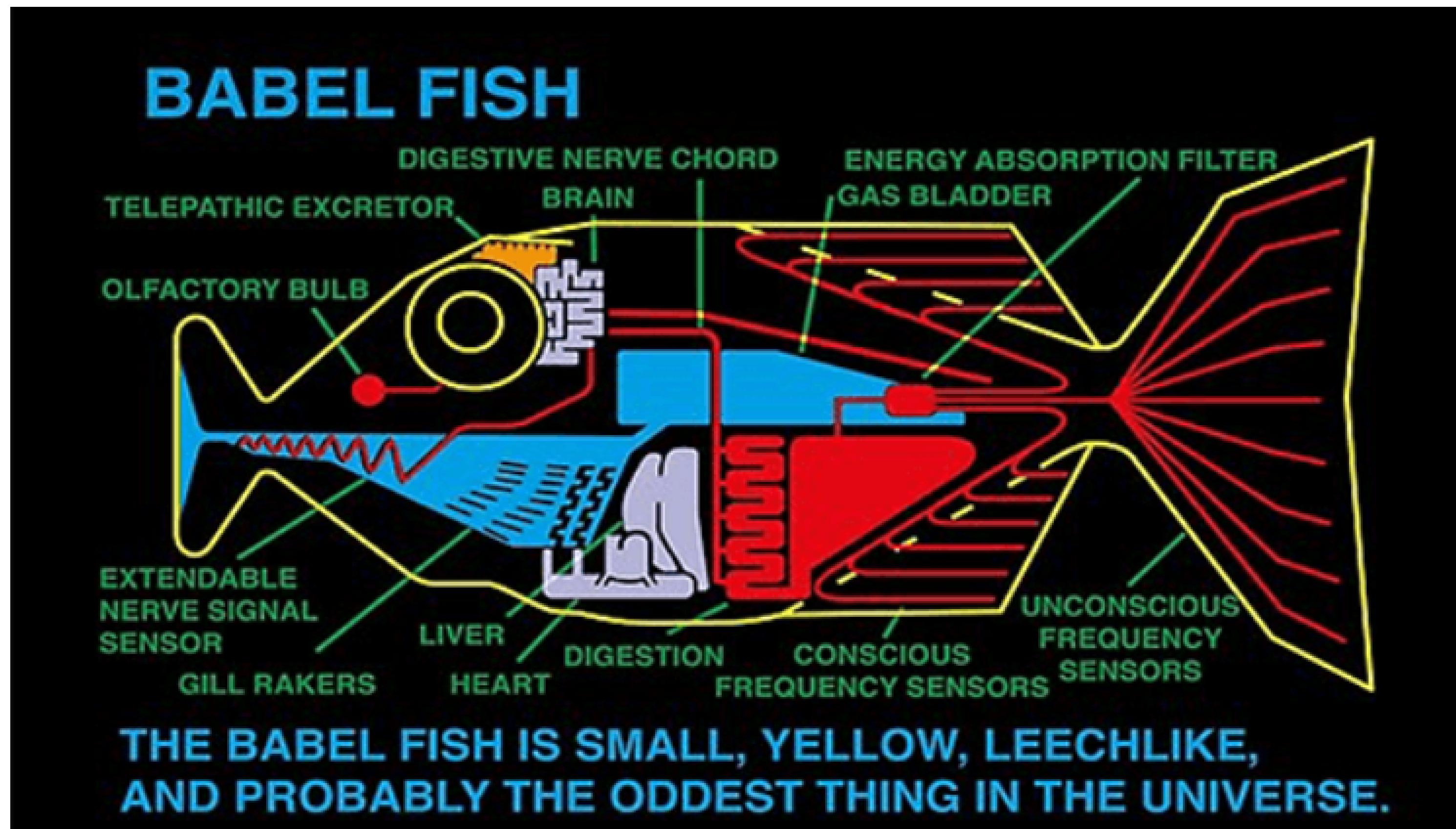


www.projectcartoon.com

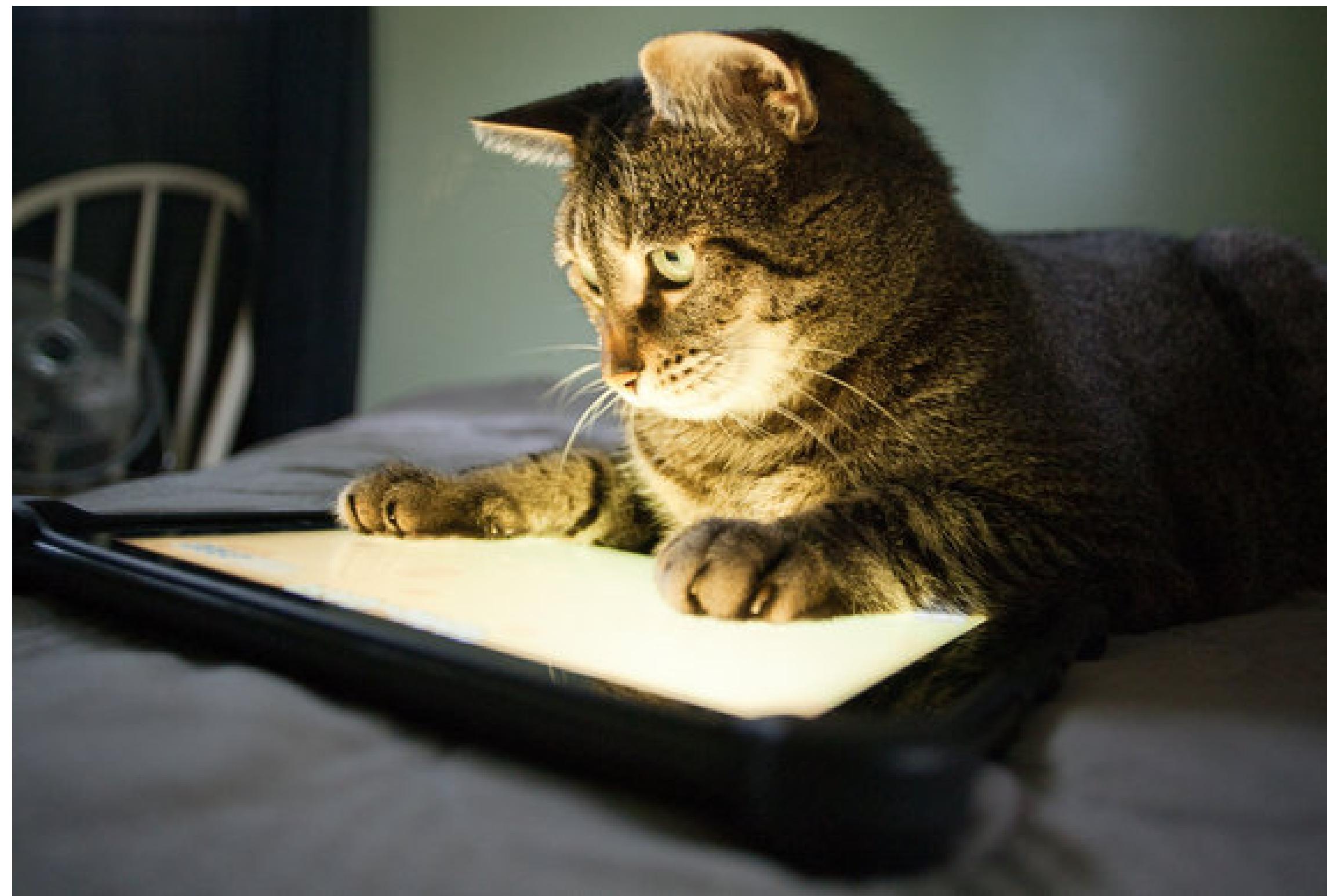
How the final product looked like

"Behaviour-driven Development polega na tworzeniu oprogramowania przez opisywanie jego zachowania, z perspektywy jego udziałowców." [D. North]

Wspólny język



Myśleć jak użytkownik



Skupienie na testach akceptacyjnych

Komunikacja

- Wszyscy uczestnicy (+ biznes) odwoływali się i myśleli o systemie w ten sam sposób
- Zmniejszenie bariery pomiędzy technicznymi i nietechnicznymi członkami zespołu
- Skupienie na testach akceptacyjnych
- Najlepsze poznanie potrzeb, celów oraz oczekiwania

Sformalizowanie języka

- **Given** - stan początkowy
- **When** - akcja
- **Then** - stan końcowy

Sformalizowanie języka

- **Feature** - funkcjonalność
- **Scenario** - scenariusz
- **And** - łączenie wielu warunków
- **Background** - akcje przed scenariuszem

Gherkin

Title: Returns and exchanges go to inventory.

As a store owner,
I want to add items back to inventory when they are returned or exchanged,
so that I can track inventory.

Scenario 1: Items returned for refund should be added to inventory.

Given a customer previously bought a black sweater from me
and I have three black sweaters in inventory,
when they return the black sweater for a refund,
then I should have four black sweaters in inventory.

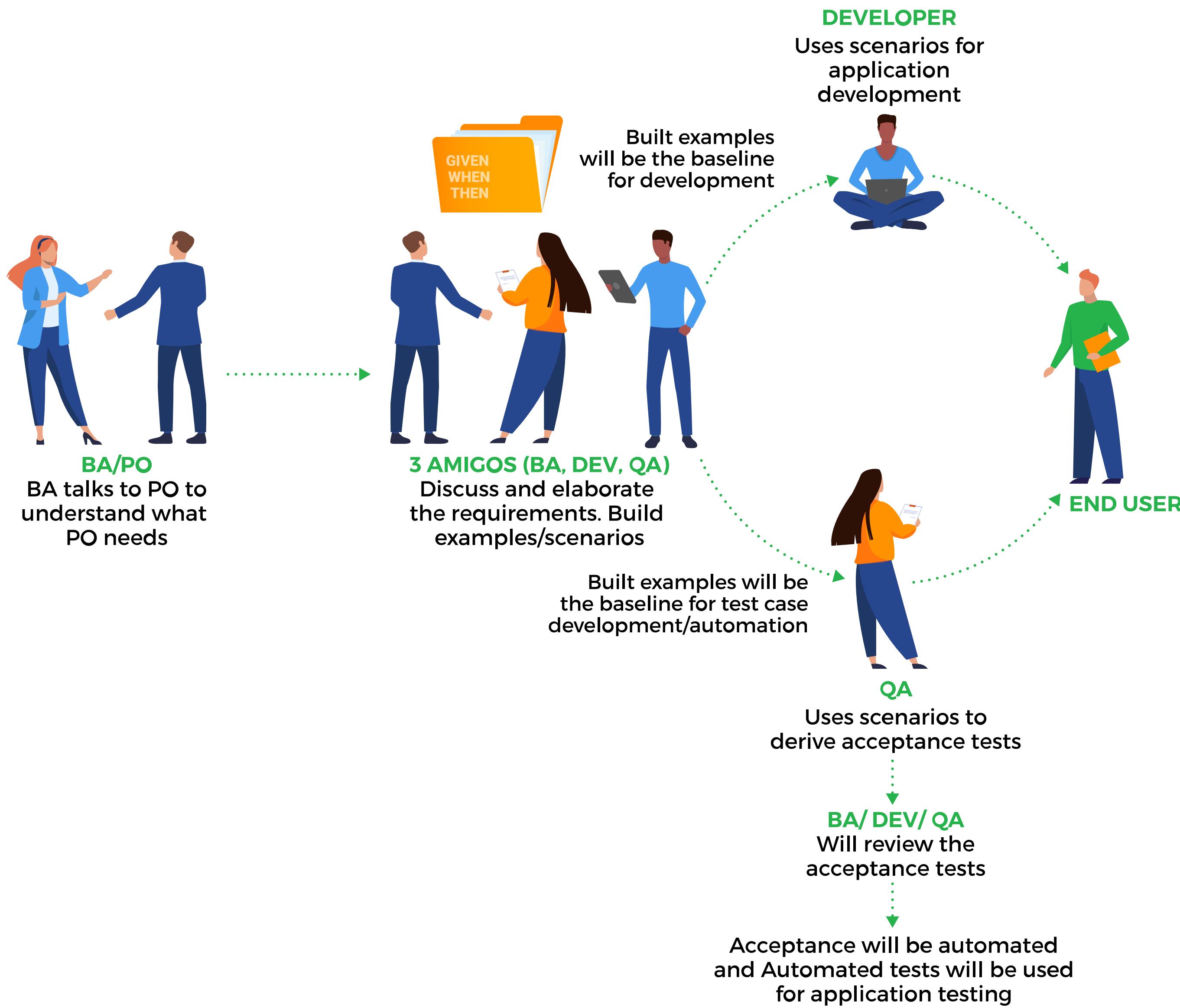
Example from [wikipedia](#)

Gherkin

- tłumaczony na kod:

```
@given("a customer previously bought a black sweater from me")
def step(context):
    pass
```

Gherkin



Gherkin

Zapraszamy do pracy nad definicjami:

- Bizness
- Designerów
- Product Mgmt/Owner

Jako narzędzie

- Jeden ze standardów w automatyzacji
- Prosta składnia – niewiele słów kluczowych
- Ustandardyzowana struktura zrozumiała dla wszystkich uczestników projektu
- Łatwiejsze zarządzanie (np. parametryzacja)
- Przydatny w analizie wymagań

Wady

- Dodatkowa rzecz do napisania
- Wymaga dużego zaangażowania i współpracy
- Wciągnąć bizness do pracy z Gharkinem może być trudne
- Źle napisane specyfikacje są kosztowne w utrzymaniu

Frameworks

- Python Behave
- JBehave
- cucumber.io
- [radish-bdd](#)
- Robot Framework
- RSpec
- [Go Ginkgo](#)

Editors

BDD vs TDD

- Podobne, z dodatkowym krokiem określenia zachowania
- Nie będziemy omawiać tego aspektu w detalach

Python Behave

A word cloud centered around the Behave framework, with words like scenario, step, behave, and passed prominently displayed. Other visible words include bdd, tags, outline, given, driven, gherkin, specify, examples, wip, failed, design, setup, steps, feature, and testing.

Feature

```
# -- FILE: features/example.feature
Feature: Showing off behave

    Scenario: Run a simple test
        Given we have behave installed
        When we implement 5 tests
        Then behave will test them for us!
```

[Behave tutorial](#)

Steps

```
# -- FILE: features/steps/example_steps.py
from behave import given, when, then, step

@given('we have behave installed')
def step_impl(context):
    pass

@when('we implement {number:d} tests')
def step_impl(context, number):
    pass

@then('behave will test them for us!')
def step_impl(context):
    pass
```

Demo

- github.com/wojciech11/se_bdd_basics

```
git clone https://github.com/wojciech11/se_bdd_basics.git
cd se_bdd_basics

# opcjonalne
python3 -m venv .venv
source .venv/bin/activate

cd zadanie_1
behave
```

Najlepsze Praktyki

- Given: present or present perfect tenses
- When: present tenses
- Then: future or "should"

Best Practices: [1](#), [2](#)

Najlepsze Praktyki

- Avoid the procedure-driven scenarios
- Declarative vs Imperative

Best Practices: [1](#), [2](#)

Ćwiczenia Praktyczne

Środowisko

- linux

Przygotowanie

Instrukcja:

https://github.com/wojciech11/se_bdd_basics

Przygotowanie

- uruchom Terminal

```
sudo apt-get install python3-venv python3-pip -y
sudo snap install atom --classic

mkdir -p workspace
cd workspace

# zauważ kropka
atom .
```

Przygotowanie - Atom

Zainstaluj plugin w Atomie (Edit->Preferences):

- cucumber
- platformio-ide-terminal

Pierwszy test - Atom

Struktura naszego projektu:

```
workspace/  
  \- example/  
    \- features/  
      |- steps/  
      \- example.feature
```

Pierwsze test

example.feature:

```
Feature: Healthy food
```

```
  Scenario: eating
```

```
    Given there were 5 cucumbers
```

```
    When I eat 3 cucumbers
```

```
    Then I should have 2 cucumbers
```

Pierwsze test

- Packages -> platformio-ide-terminal-> Toggle

```
python3 -m venv venv
source venv/bin/activate
pip install -U behave

behave -q
```

Implementacja testu

steps/food.py:

```
from behave import given, when, then, step

@given("there were 5 cucumbers")
def step_cucumber(context):
    pass

@when("I eat 3 cucumbers")
def step_eating(context):
    pass

@then("I should have 2 cucumbers")
def step_after(context):
    pass
```

Implementacja testu

```
# bez -q uruchamiam również kod  
behave
```

Kolejny scenariusz

- Na zasadzie analogi, proszę zbudować jeszcze jeden scenariusz.
- `behave -q` dla walidacji gherkin
- `behave`, jeśli dopiszenie kod python

Paremtryzacja

steps/food.py:

```
from behave import given, when, then, step

@given("there were {number:d} cucumbers")
def step_cucumber(context, number):
    pass

@when("I eat {number:d} cucumbers")
def step_eating(context, number):
    pass

@then("I should have {number:d} cucumbers")
def step_after(context, number):
    pass
```

Which Scenario do you prefer?

Scenario: Earning point test case

```
Given I open the Purchase Flights page
And I purchase a Return ticket from London to New York
And I upgrade the return flight to Business
And I complete the flight
When I log on to the Frequent Flyer site
Then I go to the account summary page
When I click on "Earned Points"
Then I see the total number of points of 1350
```

or...

Background:

```
Given the following flight points schedule:
```

From	To	Class	Points
London	New York	Economy	550
London	New York	Business	800

← Facts

```
And Stacey is a Frequent Flyer member
```

Scenario: Flights earn points based on distance travelled
and cabin class

```
Given Stacey has purchased the following tickets:
```

From	To	Class	
London	New York	Economy	
New York	London	Business	

← Past

← Present

```
When she completes the flights
```

```
Then she should earn 1350 points
```

← Future or "Should"

Can A Gherkin Scenario Be Too Simple?

Example 1

```
Scenario: Logging in with an invalid password
  Given I am on the login page
  When I enter "my.email@email.com" into the email field
  And I enter "WrongPassword" into the password field
  And I click on the Login button
  Then I should get an error message "Wrong username or password"
```

Example 2

```
Scenario: Viewing tasks
  Given I am on the home page
  When I enter a valid username and password
  Then I should go to the home page
  And I should see a list of tasks assigned to my group
  When I click on one of the tasks
  Then I should see the details of the task
  When the task is assigned to me
  Then I should be able to modify the fields in the task
```

Example 3

Scenario Outline: Analysts can view tasks in their group and modify tasks assigned to them

```
Given Alan is an Analyst in the Compliance group
And Task <Task> in group <Task Group> is assigned to <Assignee>
When he views the current list of tasks
Then he should be able to perform the following actions on the task:
| View the task in the list of tasks | <Is Visible> |
| Modify the task                 | <Is Modifiable> |
| Assign the task to himself     | <Is Assignable> |
```

Examples:

Task	Task Group	Assignee	Is Visible	Is Modifiable	Is Assignable
1	Compliance	Alan	Yes	Yes	No
2	Compliance	Joe	Yes	No	No
2	Compliance	Unassigned	Yes	No	Yes
3	Accounting	Susan	No	No	No

Zadanie 1

Card Number

Expiration

CVC

Do you have a promo code ?

I Agree with the [Terms of Use](#)

Start Subscription

<https://securionpay.com/blog/best-payment-form-ux-practices/#a7>

Zadanie 2

Jako potencjalny klient apteki www.apteka.test chcę mieć możliwość zapisać się do newslettera, aby być na bieżąco informowanym na temat aktualnych promocji apteki.

- Opis funkcjonalności
 - Na głównej stronie www.apteka.test jest pole z napisem "zostaw email" i przycisk "zapisz mnie"
 - Po podaniu poprawnego maila i kliknięciu przycisku, otwiera się nowy widok z komunikatem informującym o wysłaniu linka aktywacyjnego.
- Obsługa błędów
 - Przy próbie zapisania do newslettera bez podania adresu, pojawia się komunikat o treści “Formularz zawiera błędy. Sprawdź poprawność danych”

Zadanie 3

Scenario: Eating

Given there were <start> cucumbers

When I eat <eat> cucumbers

Then I should have <left> cucumbers

Examples:

start	eat	left	
5	2	3	
3	3	0	
0	0	0	

Zadanie 3

```
Feature: Login
  Jako użytkownik powinien mieć zalogowania, po którym widoczny jest link do pobrania pliku

  Scenario: Logowanie z poprawnymi danymi użytkownika
    When Wprowadzam prawidłową nazwę użytkownika
    And Wprowadzam poprawne hasło
    And Klikam przycisk login
    Then Link do pobrania pliku jest wyświetlony

  Scenario Outline: Logowanie z niepoprawnymi danymi użytkownika
    When Wprowadzam nieprawidłową nazwę <nazwa> i hasło <hasło> użytkownika
    And Klikam przycisk login
    Then Komunikat Nieprawidłowe dane Logowania powinien zostać wyświetlony

  Examples:
    | nazwa      | hasło          |
    | Paweł      | Ciesielskik123 |
    | Wiesław   | .              |
    | !@#$%^&*()| Paleta        |
```

Zadanie 3

- Jako użytkownik codeacademy chcę mieć możliwość filtrowania w katalogu dostępnych kursów, aby szybciej znaleźć interesujący mnie kurs.

- Opis funkcjonalności:
 - kategorie filtrów: temat, język
 - w kategorii temat dostępne opcje: Full Catalog, Web Development, Programming, Data Science, Partnerships, Design
 - w kategorii język dostępne opcje: HTML & CSS, Python, JavaScript, Java, SQL, Bash/Shell, Ruby
 - nie można wybrać języka i tematu jednocześnie
 - zaznaczonej opcji nie można odznaczyć, można wybrać tylko inną opcję
 - domyślnie zaznaczoną opcją jest Full Catalog

Referencje

- <https://dannorth.net/introducing-bdd/>
- <https://jenisys.github.io/behave.example/tutorials>

Sposób myślenia

Given-When-Then jest pomoga opisywać funkcjonalność w testach czy designie:

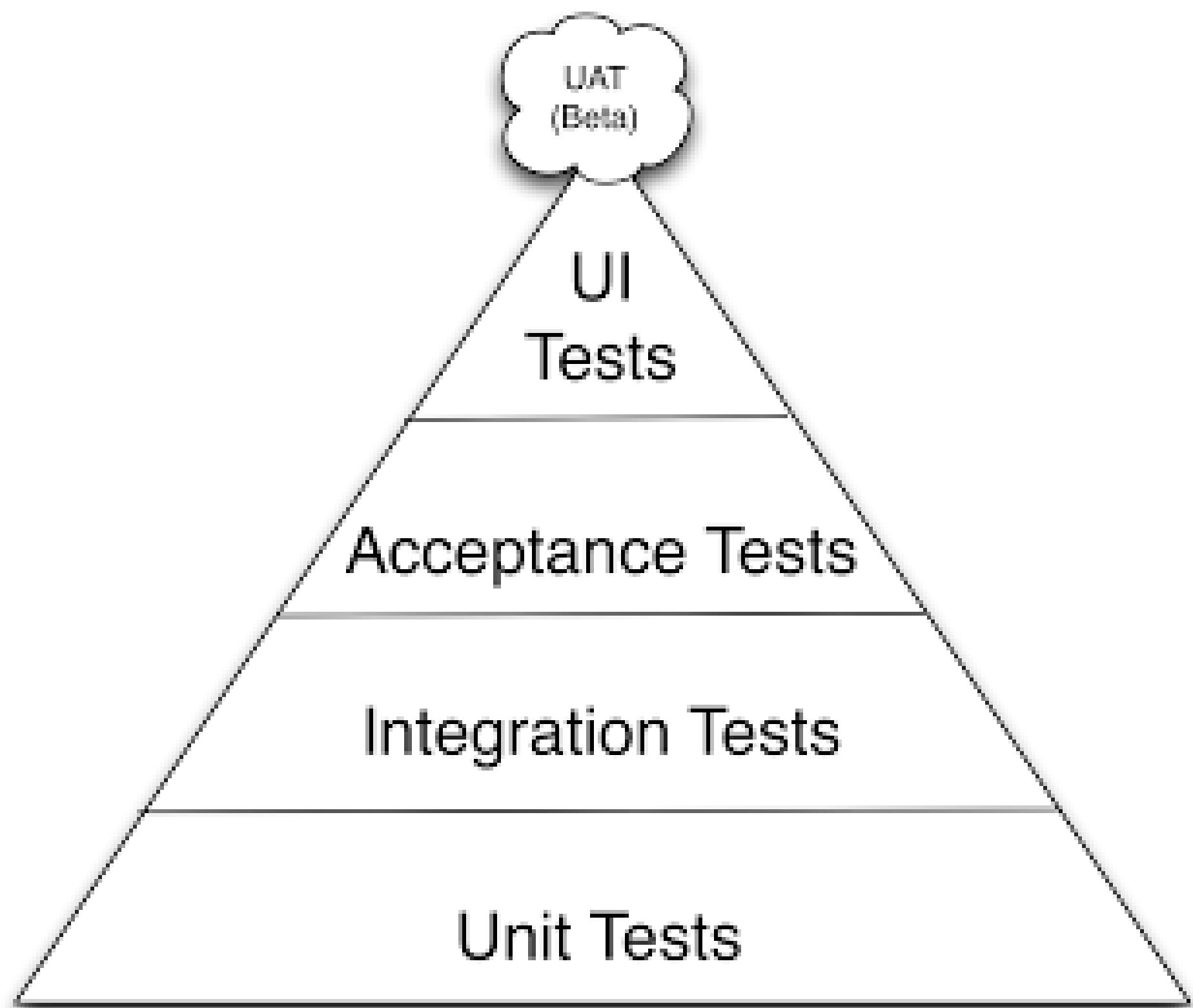
```
# Given: we are logged in
def when_pageOpened_then_counterIncrease():
    pass
```

DZIĘKUJĘ ZA UWAGĘ

Wojciech Barczyński

BACKUP SLIDES

Testy



Technologie



RobotFramework

