Programowanie internetowe

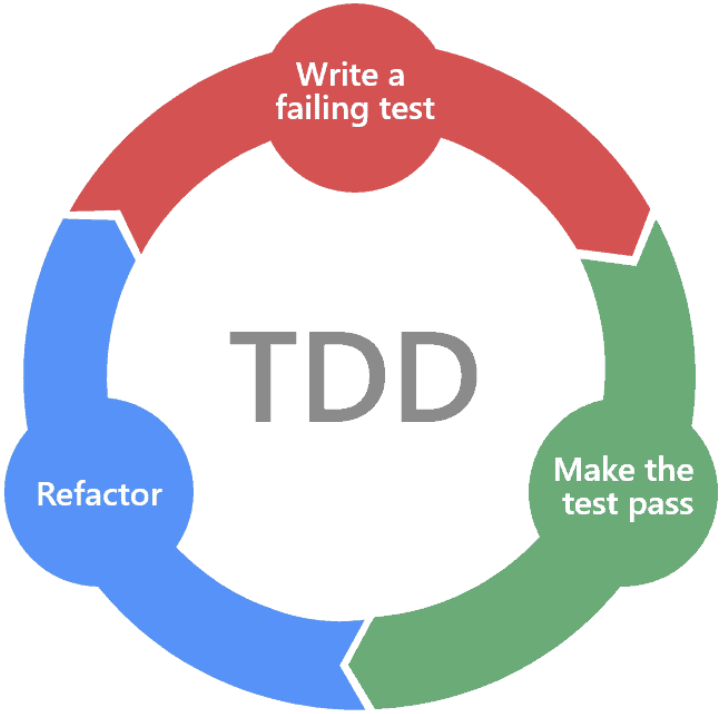
Test Driven Development

Programowanie sterowane testami

Opracował: inż. Grzegorz Petri

Przegląd zagadnień

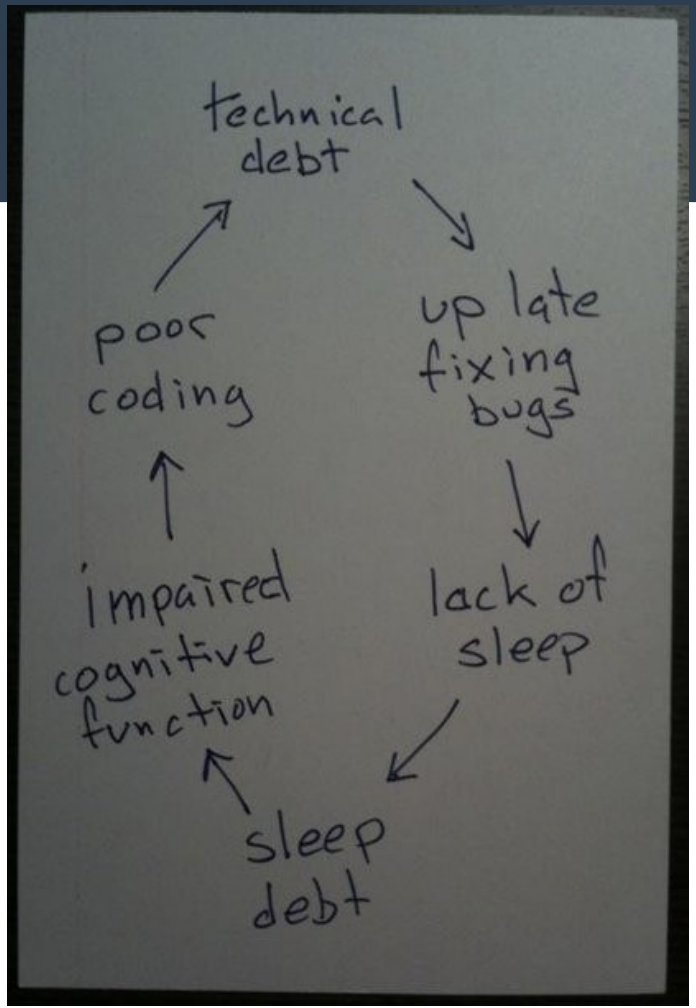
- Dług technologiczny
- Czy warto stosować TDD?
- Rodzaje testów
- „Narzędzie” RGR
- Czym jest asercja
- Przykłady
- Frameworki do TDD



Dług technologiczny

Code Debt / Design Debt / IT Debt

- Pojęcie powstałe w latach 90
- Brak projektu (*analizy*) → ciągłe przerabianie (*wymagania*) → konsekwencje (*koszty*, *czas*, *problemy i błędy*)
- Każdy projekt IT startuje z długiem (*czasem świadomie*)
- Dług techniczny jest opóźnieniem, spowAlniAniem pracy
- … jest pójściem na skróty, by zaspokoić bieżące potrzeby, i nie możliwy do spłacenia lub uniemożliwiający rozwój
- Zaciąganie długu może być celowe i świadome biorąc pod uwagę cykl wydawniczy danej aplikacji
- *Występuje niezależnie od kompetencji, czy doświadczenia*



Czas = częsta refaktoryzacja < Dług IT

Zaciągać dług, czy nie?

Wady

- ✘ Zmniejszanie produktywności oraz efektywności pracy zespołu
- ✘ Trudności w spłaceniu długu:
  - Problemy wprowadzania zmian w kodzie
  - Kosztowne i czasochłonne zmiany
  - Nowi pracownicy dziedziczą dług
- ✘ Mnogość błędów i niepełne funkcjonalności utrudniają podjęcie konkretnych działań
- ✘ Obniżenie składowych jakości produktu końcowego (*wydajność*, *niezawodność itp*)

Zalety

- ✔ Tymczasowe rozwiązanie służące weryfikacji pomysłu **Product Ownera** lub…
- ✔ … kiedy wiadomo, że powstanie nowa wersja od podstaw
- ✔ Wypuszczenie niedokończonej aplikacji, by otrzymać feedback i kierunek rozwoju od użytkownika
- ✔ Koszty/czas naprawy produktu są mniejsze, niż tworzenie testów w danym projekcie

Zdecyduj co jest lepsze dla Twojego projektu

Czym jest TDD

- Zmiana podejścia do programowania
- Najpierw powstaje test dla funkcjonalności - ta powstanie tak, by pomyślnie przeszła test
- Dodatkowy krok w procesie realizacji większych projektów owocuje efektywniejszą oraz oszczędniejszą pracą
- Jedno z narzędzi programowania (*m.in. metodyki programowania ekstremalnego*)
- *Powodem, by napisać jasne i klarowne wymagania co do projektu, które deweloper musi zrozumieć, by napisać testy.*

Nowe podejście do pracy

1. Zaprojektuj klasę / API
2. Stwórz zestaw testów
3. Zaimplementuj klasę/API
4. Przeprowadź testy
5. Wprowadź poprawki

Kodować, czy testować?

Testowanie

- ✔ Mniejsza liczba błędów w perspektywie projektu
- ✔ Symulacja i Replikacja błędów
- ✔ Naprawianie bez nowych błędów
- ✔ Nowa architektura… pracy
- ✔ Relaks i zabawa – inaczej: *pewność, że kod działa*

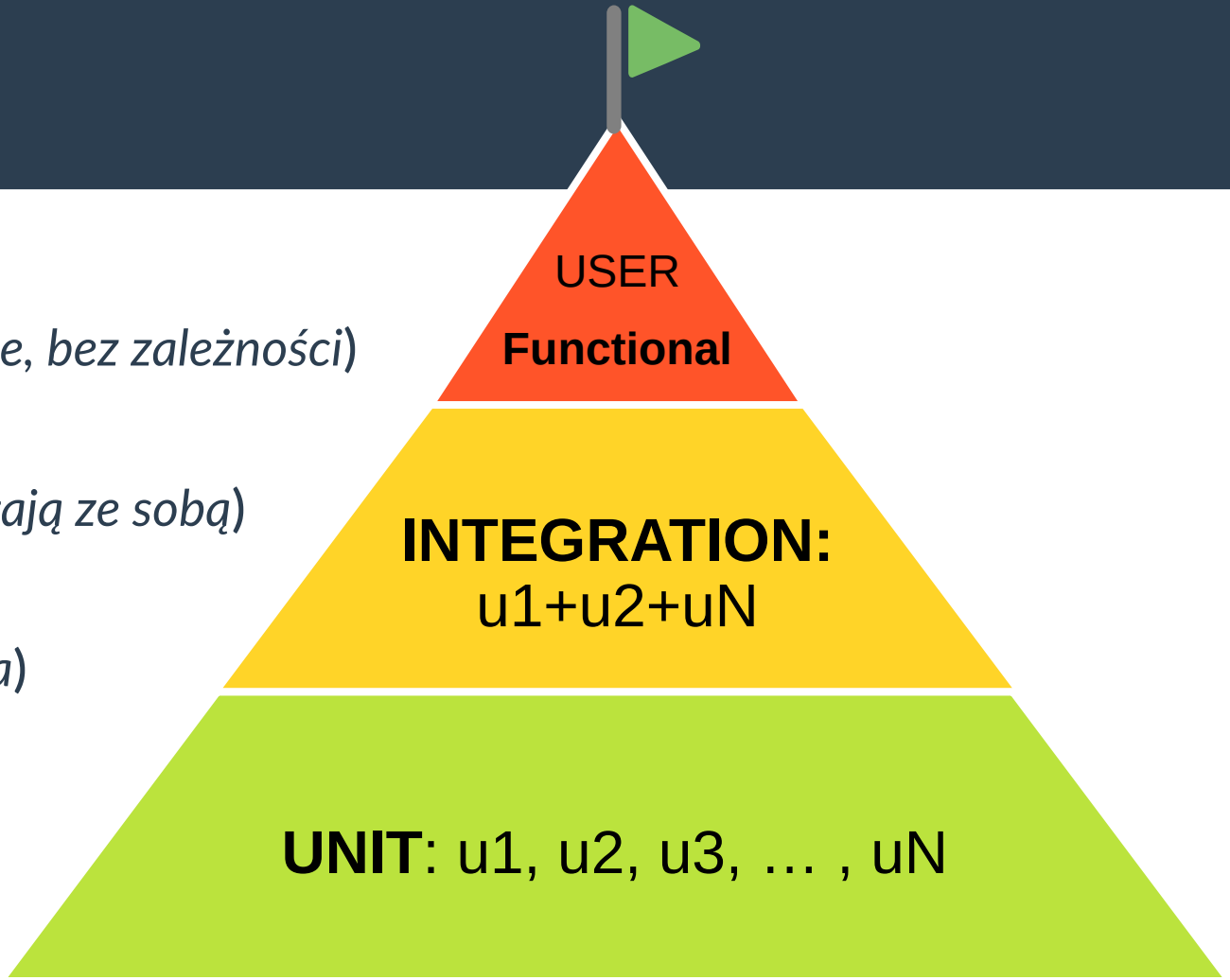
Programowanie

- ✘ Czas na naukę
- ✘ Dodatkowy krok/zależność projektu
- ✘ *Tego pana nie obsługujemy* – części systemu, których nie warto testować
- ✘ Zbyt mały/prosty projekt
- ✘ Naprawa błędów lepsza niż testy

Zdecyduj co jest lepsze dla Twojego projektu

Rodzaje testów

- **Jednostkowe (unit)**
  (*pojedyncza metoda w klasie, bez zależności*)
- **Integracyjne**
  (*czy pojedyncze części działają ze sobą*)
- **Funkcjonalne**
  (*z perspektywy użytkownika*)
- **Akceptacyjne**
  (*czy aplikacja działa tak jak chcieliśmy*)



Cykl R-G-R

- **Red** – test niezaliczony (FAIL)
- **Green** – test zaliczony (OK)
- Refactor – poprawianie kodu

Realizacja cyklu

1. Dodanie testu
2. Dostosowanie kodu do zaliczenia testu
3. Końcowa refaktoryzacja kodu

Narzędzia do testów

- **Postman** (REST API)
- **JUnit** (Java) [Link]
- **PHPUnit** (PHP) [Link]
- **QUnit** (jQuery,JS) [Link]
- **Mocha** (JavaScript) [Link]
- **Jest** (JavaScript) [Link]
- *AVA, Tape, Jasmine (JS)*

- ✔ Instalacja / integracja
- ✔ Testowanie kodu
- ✔ Testowanie API
- ✔ Biblioteki asercji
- ✔ Rozszerzenia do narzędzi
- ✔ Generowanie plików podsumowań
- ✔ Generowanie dokumentacji

Porównanie: https://stackshare.io/stackups/jest-vs-mocha-vs-qunit

PHPUnit

Narzędzia do testów

Instrukcja: https://phpunit.de/getting-started/phpunit-8.html

- **Instalacja jako:**
  - ✔ lokalna biblioteka PHAR
  - ✔ systemowa biblioteka
- **Uruchamianie testów:**
  - ✔ wybrany test / katalog testów
  - ✔ w lokalnej przeglądarce (PHP)
  - ✔ poprzez CLI / skrypty systemu
- **Widoki i informacje:**
  - ✔ Konfigurowalne komunikaty
  - ✔ Export do różnych formatów
- **Technologia:**
  - ✔ Dowolna przeglądarka W3C
  - ✔ PHP Archive

```
gp@gpl:~/Workspace/PHPUnit$ ./tools/phpunit.phar -v tests
 --testdox-html res.html --log-junit res.xml
PHPUnit 8.5.14 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.24-0ubuntu0.18.04.7

F.F.
           4 / 4 (100%)

Time: 81 ms, Memory: 10.00 MB

There were 2 failures:

1) DrugiTest::testTwoObjects
A NULL nie jest tablicą
Failed asserting that null is of type "array".

/home/gp/Workspace/PHPUnit/tests/DrugiTest.php:20

2) PierwszyTest::testGreetings
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'The Fake is a Lie'
+'The Cake is a Lie'

/home/gp/Workspace/PHPUnit/tests/PierwszyTest.php:18

FAILURES!
Tests: 4, Assertions: 7, Failures: 2.
```

PHPUnit

Eksport rezultatów

Drugi

✗ Two objects
✓ Numerek

Pierwszy

✗ Greetings
✓ Goodbye

Podany plik XML nie zawiera żadnych informacji o stylach z nim związanych. Poniżej wyświetlone jest drzewo dokumentu.

```
−<testsuites>
  −<testsuite name="tests" tests="4" assertions="7" errors="0" warnings="0" failures="2" skipped="0"
   time="0.000340">
    −<testsuite name="DrugiTest" file="/home/gp/Workspace/PHPUnit/tests/DrugiTest.php" tests="2"
     assertions="4" errors="0" warnings="0" failures="1" skipped="0" time="0.000199">
      −<testcase name="testTwoObjects" class="DrugiTest" classname="DrugiTest" file="/home
       /gp/Workspace/PHPUnit/tests/DrugiTest.php" line="10" assertions="3" time="0.000165">
        −<failure type="PHPUnit\Framework\ExpectationFailedException">
            DrugiTest::testTwoObjects A NULL nie jest tablicą Failed asserting that null is of type "array".
            /home/gp/Workspace/PHPUnit/tests/DrugiTest.php:20
          </failure>
        </testcase>
        <testcase name="testNumerek" class="DrugiTest" classname="DrugiTest" file="/home/gp/Workspace
       /PHPUnit/tests/DrugiTest.php" line="25" assertions="1" time="0.000034"/>
      </testsuite>
    −<testsuite name="PierwszyTest" file="/home/gp/Workspace/PHPUnit/tests/PierwszyTest.php" tests="2"
     assertions="3" errors="0" warnings="0" failures="1" skipped="0" time="0.000141">
      −<testcase name="testGreetings" class="PierwszyTest" classname="PierwszyTest" file="/home
       /gp/Workspace/PHPUnit/tests/PierwszyTest.php" line="10" assertions="2" time="0.000105">
        −<failure type="PHPUnit\Framework\ExpectationFailedException">
            PierwszyTest::testGreetings Failed asserting that two strings are equal. --- Expected +++ Actual @@
            @@ -'The Fake is a Lie' +'The Cake is a Lie' /home/gp/Workspace/PHPUnit/tests/PierwszyTest.php:18
          </failure>
        </testcase>
        <testcase name="testGoodbye" class="PierwszyTest" classname="PierwszyTest" file="/home
       /gp/Workspace/PHPUnit/tests/PierwszyTest.php" line="21" assertions="1" time="0.000036"/>
      </testsuite>
    </testsuite>
  </testsuites>
```

PHPUnit

15 z 60 dostępnych asercji

Dokumentacja: *https://phpunit.readthedocs.io/en/9.5/assertions.html*

assertEquals()
assertEqualsIgnoringCase()
assertEqualsObjectEquals()
assertFileEquals()
assertFileExists()
assertFileIsReadable() / writeable
assertTrue() / assertFalse()
assertNull() / assertEmpty()
assertThat() / assertContains()
assertCount() / assertIsIterable()

PHPUnit

10 z 60 dostępnych asercji

Dokumentacja: *https://phpunit.readthedocs.io/en/9.5/assertions.html*

assertInstanceOf()
assertIsArray()
assertBool()
assertIsFloat()
assertIsInt()
assertIsNumeric()
assertIsObject()
assertIsResource()
assertIsScalar()
assertIsString()

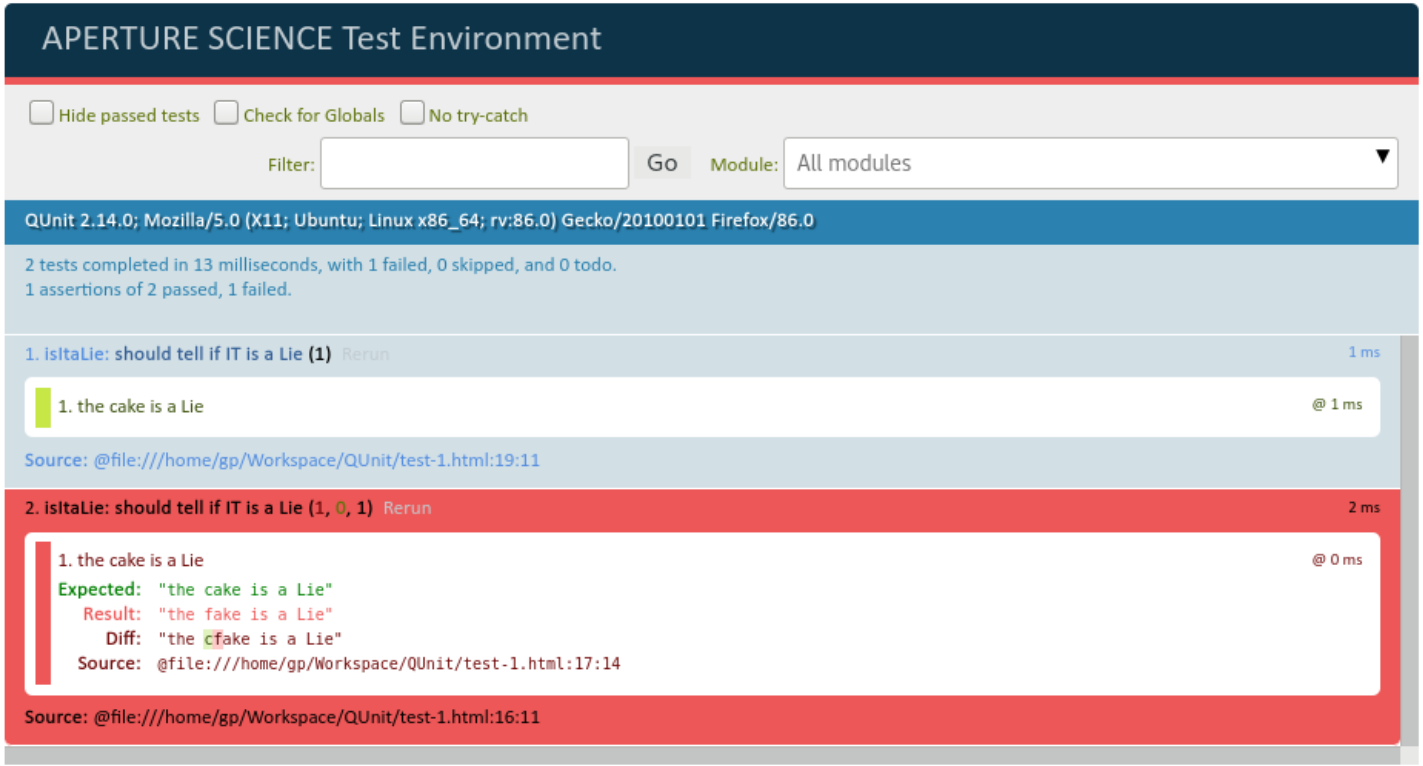
QUnit

Narzędzia do testów

QUnit

Instrukcja: https://qunitjs.com/intro/

- **Instalacja jako:**
  - ✔ lokalna biblioteka + HTML
  - ✔ zdalna biblioteka + HTML
- **Uruchamianie testów w:**
  - ✔ lokalnej przeglądarce
  - ✔ środowisku Node.js
- **Widoki i informacje:**
  - ✔ Liczba testów i czas wykonania
  - ✔ Opis testu i jego szczegóły
  - ✔ Filtry, ukrywanie zdanych
- **Technologia:**
  - ✔ Framework bazuje na jQuery*
  - ✔ Dowolna przeglądarka W3C
  - ✔ Notacja strzałkowa oraz funkcje anonimowe



QUnit

10 z 20 dostępnych asercji

Dokumentacja: *https://api.qunitjs.com/assert/*

assert.equal()	Porównanie nieścisłe – tylko wartości
assert.deepEqual()	Porównanie rekursywne
assert.propEqual()	Porównanie ścisłe właściwości obiektu - typu i wartości
assert.strictEqual()	Porównanie ścisłe typu i wartości
assert.true()	Porównanie boolowskie – 1-szy argument musi być TRUE, aby zaliczyć test
assert.false()	Porównanie boolowskie – 1-szy argument musi być FALSE, aby zaliczyć test
assert.ok()	Sprawdza, czy pierwszy argument jest prawdziwy
assert.step()	Sprawdza postęp wykonywanych testów zwracając komunikat
assert.verifySteps()	Porównuje listę kroków (etykiet tekstowych) sprawdzając kolejność i wartości dostarczone poprzez assert.step()
assert.pushResult()	Report the result of a custom assertion.

QUnit

20 z 20 dostępnych asercji

Dokumentacja: *https://api.qunitjs.com/assert/*

assert.async()	Instruct QUnit to wait for an asynchronous operation.
assert.timeout()	Set the length of time to wait for async operations before failing the test.
assert.expect()	Specify how many assertions are expected to run within a test.
assert.notDeepEqual()	An inverted deep recursive comparison.
assert.notEqual()	A non-strict comparison, checking for inequality.
assert.notOk()	Check if the first argument is falsy.
assert.notPropEqual()	A strict comparison of an object's own properties, checking for inequality.
assert.notStrictEqual()	A strict comparison, checking for inequality.
assert.rejects()	Test if the provided promise rejects.
assert.throws()	Test if a callback throws an exception.


assertThat(question);