Paweł Rajba
pawel@cs.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Podstawy testowania

Agenda

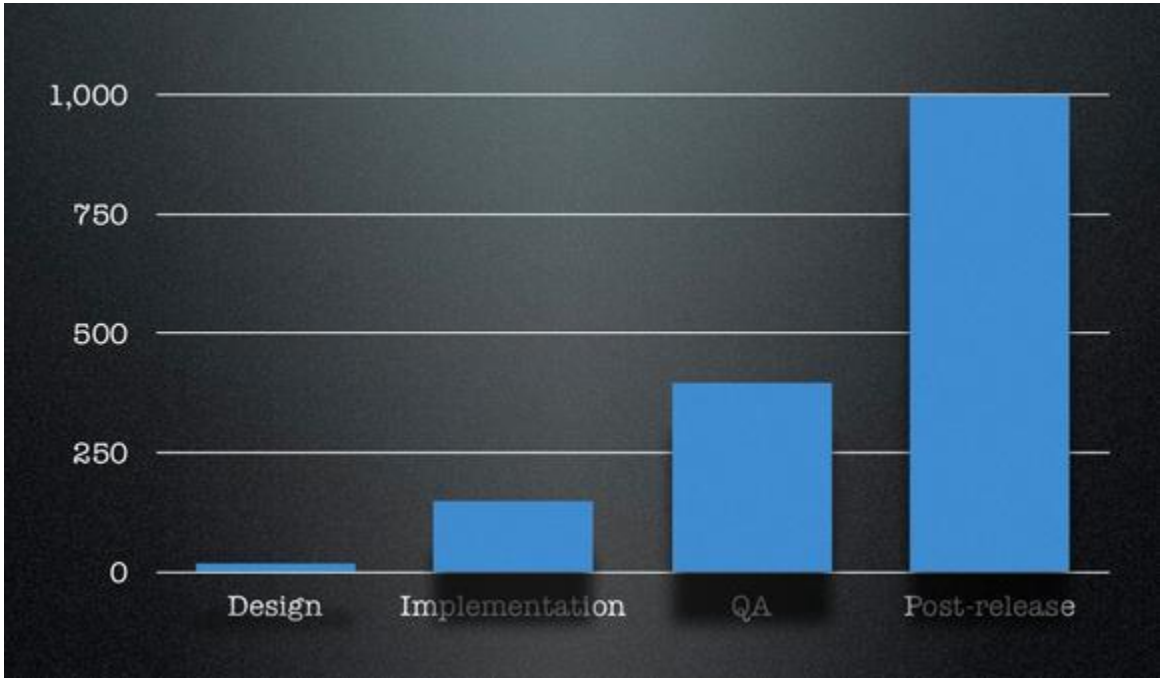
- Wprowadzenie
- Czym wspierać testowanie?
- Rodzaje testów?
- Co testujemy?
- Kiedy testować?
- Jak budować testowalne systemu?
- Moq

Wprowadzenie

- Debuggowanie i testowanie to nie to samo
- Testowanie to systematyczne próby „zepsucia” aplikacji
- Programiści nie są przyzwyczajeni do poświęcania 40% swojego czasu na testy
  - Dla krytycznych projektów (np. kontrola lotów, monitorowanie reaktorów) testy mogą kosztować 5 razy więcej niż wytworzenie samego kodu

Wprowadzenie

- Czas potrzebny na poprawę błędu



http://www.slideshare.net/Skud/test-driven-development-tutorial

Wprowadzenie

- Co daje testowanie?
  - Mniej błędów w kodzie produkcyjnym
  - Wykrycie błędów, których nikt się nie spodziewał
  - Automatyczne testy pozwalają na szybkie wykrycie błędów i niezgodności
  - Upewnienie się, że system działa zgodnie ze specyfikacją
  - Utrzymywanie poprawnie działającego kodu
    - Gdy ktoś coś zmieni, łatwo wykryć, czy dalej jest ok
  - Lepszy design

Czym wspierać testowanie?

- Dummy
  - Np. napis przekazywany do metody
- Fake
  - Np. sztuczna implementacja BD oparta na kolekcjach
- Stub
  - Obiekt, którego uczymy konkretnych zachowań
- Mock
  - Zaprogramowany „gotowiec” do udawania

Rodzaje testów

- Jednostkowe
- Integracyjne
- Wydajnościowe
- Obciążeniowe
- Interfejsów
- Bezpieczeństwa
- Akceptacyjne

Co testujemy?

- Stan (state verification)
- Zachowanie (behaviour verification)

Kiedy testować?

- Testy szybkie (np. jednostkowe)
  - Po każdym „check-in”
- Testy wolne (np. integracyjne)
  - Np. raz dziennie
- Continuous Integration

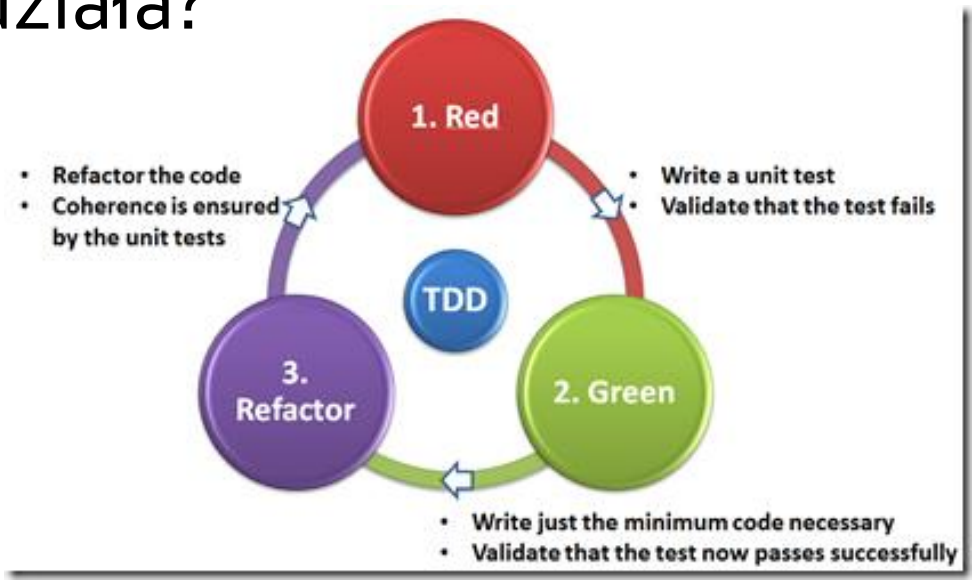
Jak budować testowalne sytemy?

- Zgubne zależności
  - Mamy warstwy UI, BL, DAL → jak testować?

- Izolacja testowanych jednostek
  - Znaczenie interfejsów
- Struktura projektu
  - Wzorzec „objects mother”
  - Rozkład projektów
- Znaczenie automatyzacji testów

Test Driven Development

- Najpierw myślimy o użyciu, potem o kodowaniu
  - Czyli najpierw testy, potem implementacja
- Jak to działa?



Ważne są krótkie cykle

http://elstarit.nl/wp-content/uploads/2014/08/UNIT_TEST_TDD.png

Test Driven Development

- Dlaczego warto?
  - Nie ma nieużywanego kodu
  - Kod wspiera wymagania klienta
  - Dobre pokrycie kodu testami
    - Łatwiejsze utrzymanie
  - Pewna forma dokumentacji (tzw. executable documentation)
- Warto zobaczyć:
  - http://www.slideshare.net/guestc8093a6/test-driven-development-1000421
  - http://www.slideshare.net/Skud/test-driven-development-tutorial

Moq

- Oglądamy przykłady na
  - https://github.com/Moq/moq4/wiki/Quickstart
- Przeglądamy testy z DDDSample