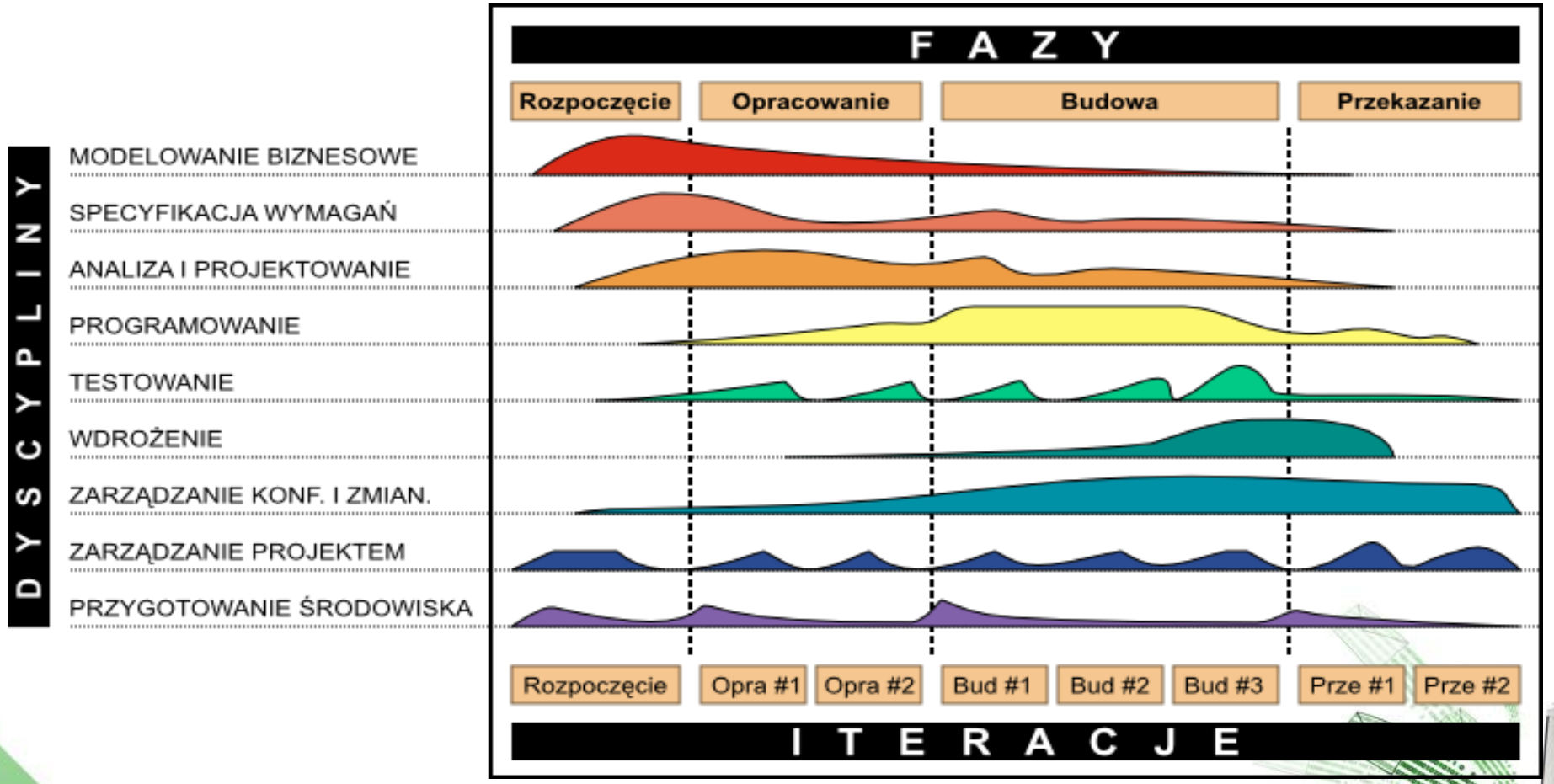
Jakub Szatkowski

Testowanie

Plan prezentacji

- Po co testować?
- Rodzaje testów i różnice między nimi
- Testy jednostkowe
- Praktyka NUnit

Proces wytwórczy



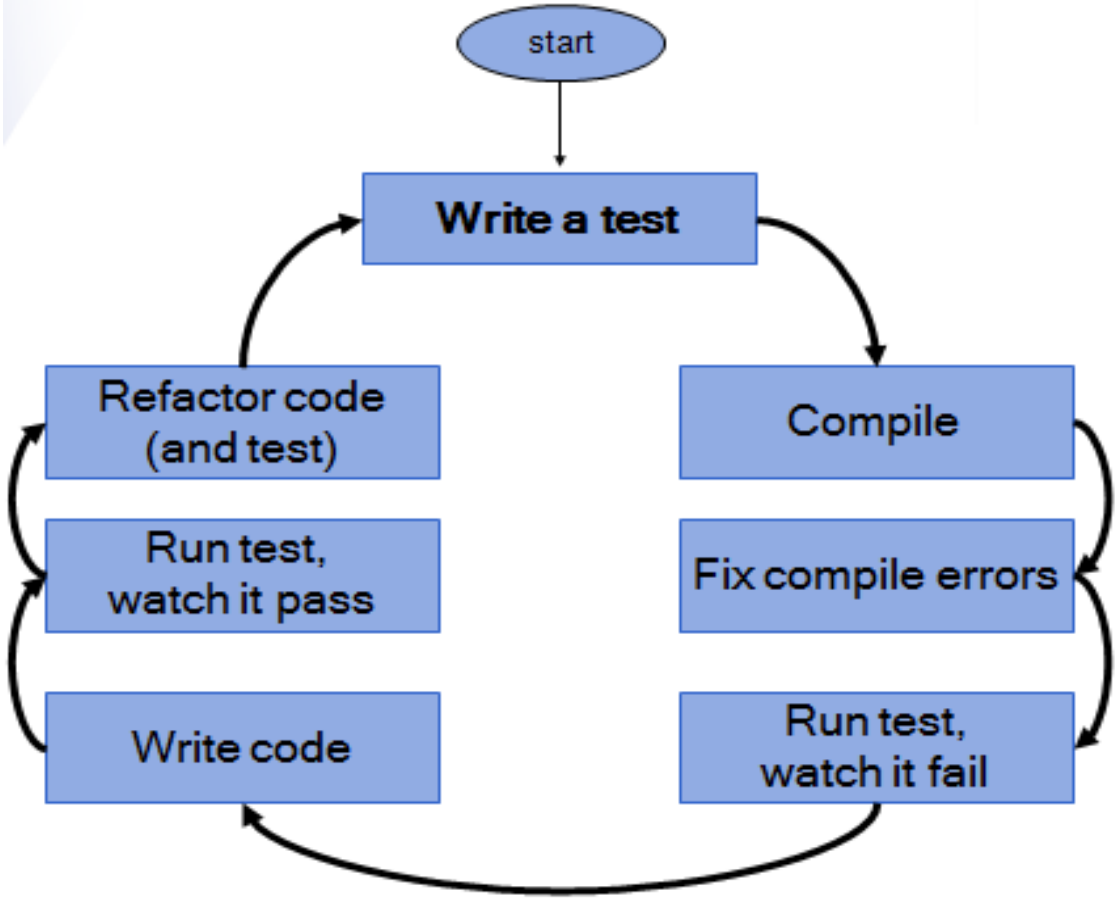
Korzyści

- Wykrywanie błędów
- Szybka weryfikacja kodu po modyfikacji
- Sprawdzenie działania poszczególnych funkcji
- Weryfikacja założonych wymagań funkcjonalnych i niefunkcjonalnych
- **Zmniejszenie kosztów produkcji**

Testy (nie)dobre na wszystko

- Nigdy nie mamy 100% pewności, że po udanych testach oprogramowanie nie zawiera błędów
- Testy nie udowadniają, że kod jest poprawny (do tego służą metody dowodzenia poprawności oprogramowania) tylko pozwalają wykryć określone błędy
- Nigdy nie sprawdzimy oprogramowania dla wszystkich możliwych danych testowych
- Nigdy nie sprawdzimy oprogramowania odtwarzając wszystkie możliwe czynności użytkownika (dotyczy aplikacji z GUI)

Test Driven Development



Czarna i biała skrzynka

Czarna skrzynka

- Testy funkcjonalne
- Wcielamy się w rolę użytkownika bez zagłębiania się w szczegóły
- Najlepiej angażować do tych testów osoby nie mające nic wspólnego z wytwarzaniem kodu

Biała skrzynka

- Testy strukturalne
- Tester analizuje poszczególne moduły kodu (mając do niego dostęp)

Podział ze względu na sposób wykonywania

- Manualne
- Automatyczne

TESTY	Dane automatyczne	Dane ręczne
Wykonanie automatyczne	Rzadziej stosowane	Często wykonywane
Wykonanie ręczne	W praktyce niespotykane	Często wykonywane

Podział ze względu na zakres

- Testy jednostkowe (testowanie na najbardziej podstawowym poziomie, działanie pojedynczych funkcjonalności)
- Testy systemowe (bezpieczeństwa, wydajnościowe – wymagania niefunkcjonalne, aplikacja jako całość)
- Testy integracyjne (współpraca komponentów)

Testy jednostkowe - struktura

- Model A/A/A
- Arrange – przygotowania przed właściwym tekstem, inicjalizacja zmiennych, przygotowanie danych testowych
- Act – test właściwy (wykonanie metody, którą testujemy)
- Assert – sprawdzenie wyników testu

Rodzaje testów

Metoda do testowania: int Add(int,int) klasy Math2 która dodaje dwie liczby do siebie

```
[TestMethod]
public void TestAdd()
{
    //arrange
    int a=10,b=5,expectedResult=15;
    Math2 m=new Math2();
    //act
    int actualResult=m.Add(a,b);
    //assert
    Assert.AreEqual(expectedResult,actualResult);
}
```

Podział testów jednostkowych

- Analiza ścieżek:

Określamy punkt początkowy i końcowy i badamy przebieg możliwych ścieżek. Z powodu ścieżek w pętli nie mamy możliwości analizy wszystkich ścieżek.

- Użycie klas równoważności

Wykonujemy test z kilkoma elementami zbioru o tym samym sposobie przetwarzania danych.

Klasy poprawności (dane poprawne), klasy niepoprawności (dane niepoprawne).

- Testowanie wartości brzegowych

Wewnątrz, pomiędzy lub tuż przy granicy klasy równoważności