Testowanie

Test-driven development

Plan

- Kent Beck,
- Koncepcja TDD,
- Cykl pracy.

Koncepcja

- TDD = Test-driven development,
- Proces tworzenia oprogramowania sterowany testami,
- Jednostkowe testy automatyczne.

Koncepcja

- Testy są tworzone przed napisaniem właściwego kodu,
- Automatyczne testy są nieodłącznym elementem projektu,
- Poprawne wykonanie zbioru testów zapewnia właściwe zachowanie systemu, wraz z ewolucją systemu.

Cykl

- Dodaj test,
- Uruchom testy i sprawdź, czy nowy test nie przechodzi,
- Napisz fragment właściwego kodu, który spełnia test.

Cykl

- Uruchom testy i sprawdź, czy wszystkie testy przechodzą,
- Refaktoryzuj,
- Powtórz.

Dodanie testu

- Nowo dodany test nie może przejść,
- Jeśli test przechodzi:
  - Nowo implementowana funkcja już jest w systemie,
  - Test jest źle napisany i np. zawsze przechodzi, nie testując istotnych elementów.

Kod

- Implementacja dodawanej funkcji powinna po prostu spełniać test,
- Stworzony kod nie musi być idealny, z uwagi na to, że kolejne kroki są przeznaczone do jego poprawy.

Sprawdzenie

- Utworzony kod musi przejść testy,
- Gwarantuje to, że stworzony kawałek kodu spełnia założone wymagania,
- Jest to dobry punkt startowy do refaktoryzacji.

Refaktoryzacja

- Stworzony kod powinien zostać uporządkowany,
- Gotowy i działający zestaw testów zapewnia, że refaktoryzacja nie zepsuła oprogramowania,
- Jest to idealny moment na usunięcie duplikacji.

Czy to działa?

Podsumowanie

- Metoda małych kroków,
- **Małe, zarządzalne, testowalne jednostki kodu,**
- Odpowiedni poziom testowania,
- Kent Beck, Test-Driven Development By Example, Addison-Wesley, 2008.

Q&A