Test Driven Development (TDD)

Il Test Driven Development è un processo di sviluppo del software in cui lo sviluppo vero e proprio è preceduto (e guidato, driven) dalla stesura di test automatici.

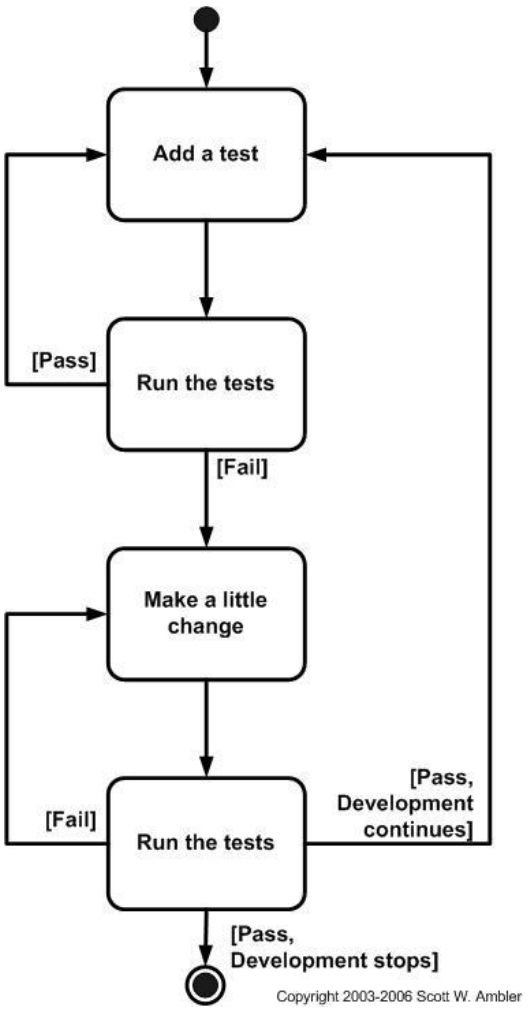
Il processo si articola sulla ripetizione di brevi cicli di sviluppo e collaudo (noti come "cicli TDD", TDD cycles) suddivisi in tre fasi successive, sintetizzate dal motto "Red-Green-Refactor".

• nella prima ("Red"), il programmatore scrive un test automatico (che necessariamente fallisce) per la funzionalità da sviluppare.

• nella seconda ("Green"), il programmatore scrive la quantità minima di codice necessaria per ottenere il superamento del test.

• nella terza, il programmatore ristruttura il codice (ovvero ne fa il refactoring).

I colori "rosso" e "verde" si riferiscono alla rappresentazione grafica di fallimento e successo di un test automatico più diffusa negli IDE.



29/01/2014 author: Donatella Abate

Behavioural-driven development (BDD)

BDD rappresenta una sintesi e un perfezionamento delle pratiche derivanti dalla TDD (Test Driven Development) e ATDD (Acceptance Test Driven Development) attuando le seguenti operazioni:

- applicare il principio "Five Why's" ad ogni storia utente proposta, in modo che il suo scopo è chiaramente legato ai risultati di business
- pensare "from the outside in", ossia implementare solo quei comportamenti che contribuiscono in modo diretto a questi risultati di business, al fine da minimizzare gli sprechi
- descrivere comportamenti in una singola notazione che è direttamente accessibile al dominio esperti, tester e sviluppatori, in modo da migliorare la comunicazione
- applicare queste tecniche fino ai più bassi livelli di astrazione del software, con particolare attenzione alla distribuzione di comportamento.

Lo sviluppo Behavior-driven è un insieme di pratiche che permettono di sviluppare software che si concentrano sugli utenti, ossia su come l'**utente** vuole che l'applicazione si comporti.

Invece di **un writing tests** si dovrebbe pensare di **specificare il comportamento (specifying behavior)** .

Si consideri un Unit test tradizionale:

```
class UserTest < Test::Unit::TestCase
  def test_name_set
    user = User.new "Audrey"
    assert_equal(user.name, "Audrey")
  end
end
```

come si può notare:

- Non sono specificati i requisiti
- La sintassi è comprensibile, ma artificiale
- Il nome del test non precisa cosa si verifica realmente.

Nello sviluppo del BDD si specifica il comportamento con frasi intere:

```
describe User do
  it "lets me assign a name" do
    user = User.new "Paul"
    user.name.should == "Paul"
  end
end
```

In questo caso ancora non sono evidenti le intenzioni degli utenti.

Dan North ha suggerito un modello che consente di descrivere le caratteristiche in linguaggio naturale. Ogni storia ha un titolo e una breve descrizione di ciò di cui la storia parla.

Il formato di ogni storia è il seguente:

1. **In order to (beneficio)**
2. **As (cosa l'utente sta sviluppando)**
3. **I want (caratteristica)**

Questa descrizione è seguita da un elenco di scenari:

1. **Given** (quanto è successo prima)
2. **When** (azioni dell'utente)
3. **Then** (risultato atteso dall'utente)

1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

A questo punto occorre:

- Scoprire la **feature** più importante
- All'interno della funzione selezionare lo scenario più rilevante per l'utente.

La seguente immagine illustra un diagramma di attività UML che mostra come ATDD e sviluppatore TDD possono lavorare insieme. Idealmente, si può scrivere un singolo test di accettazione, poi per l'attuazione del codice di produzione necessario per adempiere a tale prova prendo un approccio di sviluppo TDD. Questo a sua volta richiede di iterare più volte attraverso la scrittura di un test, del codice di produzione.