

TDD

DEVELOPPEMENT PILOTE PAR LES TESTS

Claudie FOURNEAU & Sylvain LEZIER



Remerciements



Développement Piloté par les Tests

Procédé

permettant le changement de spécifications
contrôlant la non régression fonctionnelle

Focalise l'esprit

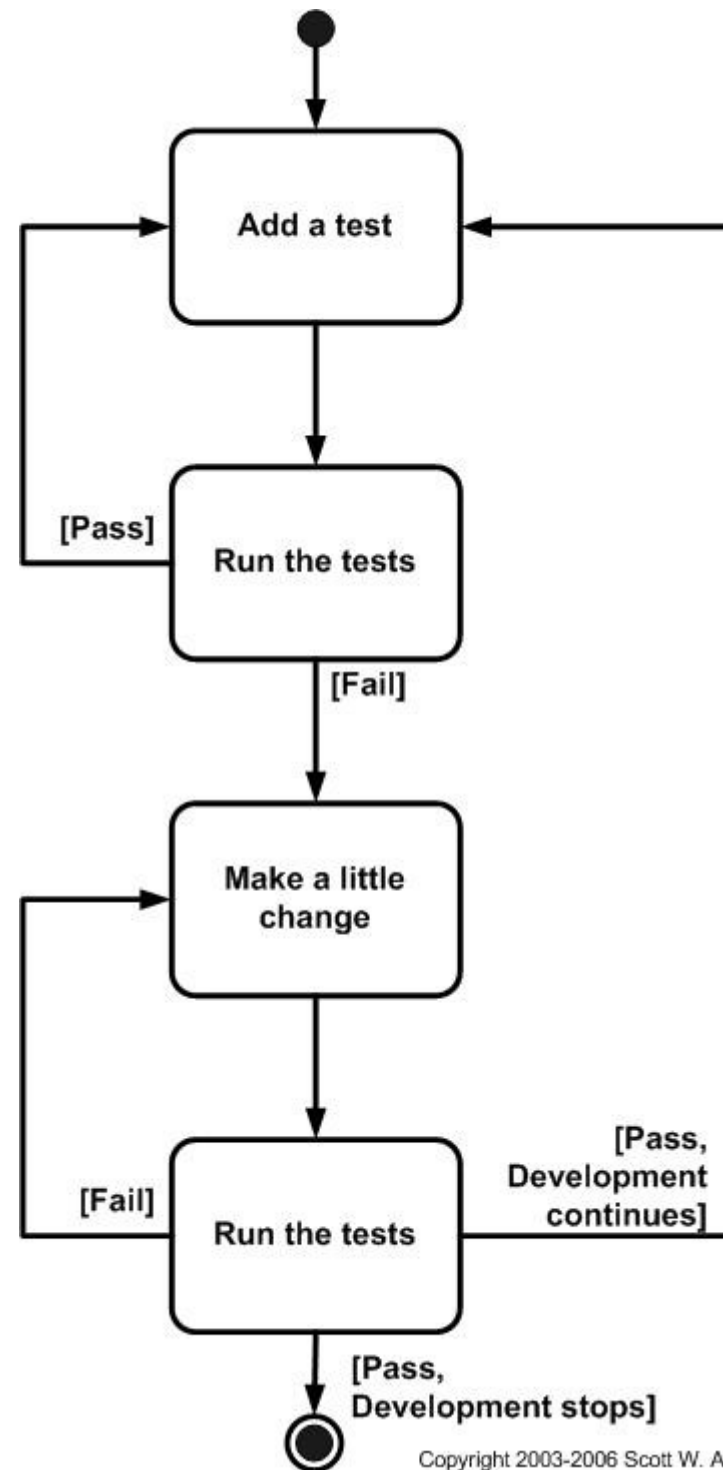
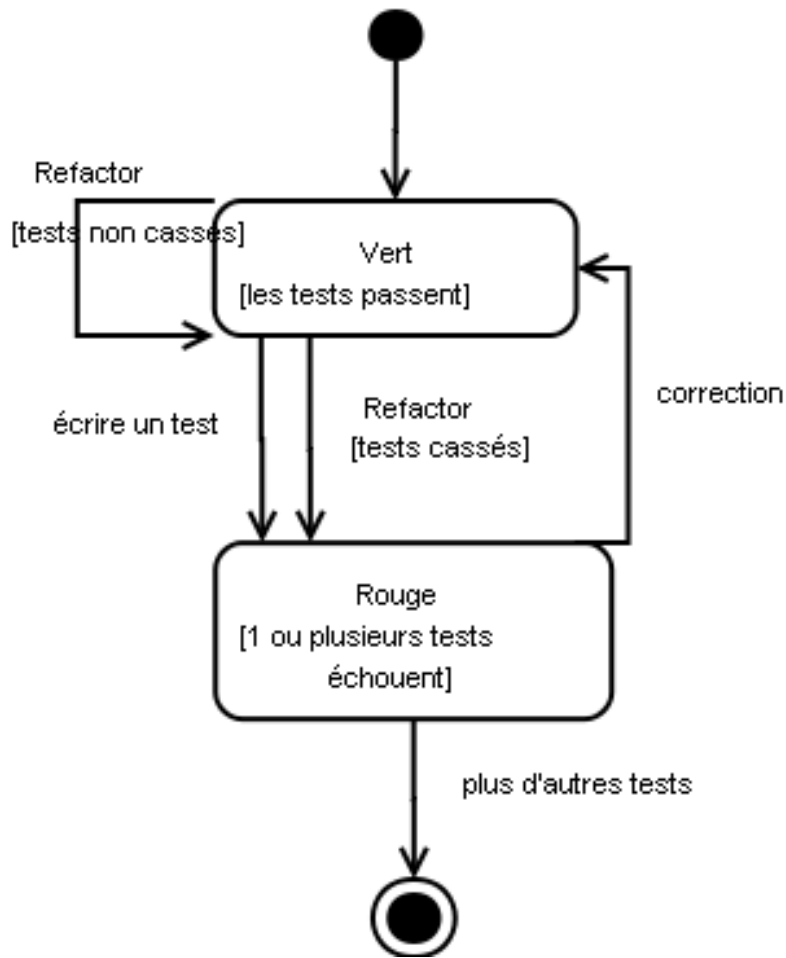
ne délivre que ce qui est nécessaire

Le système fait exactement ce qu'il doit faire et
pas plus (YAGNI)

Le système reste le plus simple possible (KISS)



Le Cycle



Bonnes pratiques

- Tests Petits
- Tests Isolés
- Fonctionnant sous n'importe quel environnement
- Tests Complets
- Tests Répétables
- Tests automatiques
- Tests expressifs
- Concevoir les tests comme des spécifications pas comme des vérifications



Bénéfices

- Filet de sécurité
- Confiance
- Documentation
- Minimise l'effort
- Feedback rapide



MISE EN PRATIQUE



FIZZ BUZZ

Ecrire un programme qui affiche les nombres de 1 à 100 sauf :

pour les multiples de 3, afficher « Fizz »

pour les multiples de 5, afficher « Buzz »

pour les multiples de 3 et 5, afficher « FizzBuzz »



UN CHAMPIONNAT DE FOOT



Les objets de Test

Simulent le comportement d'objets inutiles aux tests

dummy (bouffon)



stub (bouchon)



spy (espion)



mock (simulacre)



fake (simulateur)



Dummy (bouffon)



Objet trivial dont le comportement n'a aucun intérêt pour le test.

Implémentation classique :

Null Object : même signature que l'objet mais ne fait rien.

```
public class DummyTomTom extends TomTom {  
    @Override public void changeLaVoix(String nomDeLaVoix) {}  
}
```



Stub (bouchon)



Objet retournant toujours une même réponse

Implémentation classique :

Surdéfinition d'une méthode permettant de contrôler les retours

```
public class StubTomTom extends TomTom{  
    @Override public void ouSuisJe() { return "ici"; }  
}
```



Spy (espion)



Enregistre un appel de l'objet

(les vérifications sont faites après l'invocation).

Implémentation classique :

Surdéfinition d'une méthode.

Appel enregistré dans un champ de la classe accessible au test.

```
public class SpyTomTom extends TomTom{
    String voixDemandee;
    @Override public void changeLaVoix(String nomDeLaVoix) {
        voixDemandee = nomDeLaVoix;
    }
}
```



Mock (simulacre)



Attends un appel de l'objet

(les vérifications sont définies avant l'invocation).

Implémentation classique :

Surdéfinition des méthodes avec vérifications des paramètres lors de l'invocation.

```
public class MockTomTom extends TomTom {
    @Override public void changeLaVoix(String nomDeLaVoix) {
        assertEquals( "chewbacca", nomDeLaVoix);
    }
}
```



Fake (simulateur)



C'est une implémentation légère souvent partielle d'un comportement.

Exemples :

Client/Serveur JMS

Serveur Mail

Serveur Http

Simulateur de signalisation téléphonique



Les Types de Test

développement : Tests unitaires (JUnit).

spécification : Tests fonctionnels (Fitnesse).

performance : Tests de charge (OpenSTA).

maîtriser les besoins : Tests d'acceptance.



Intégration du Legacy

Dur et douloureux mais faisable (COURAGE).

Période de refactoring boucherie nécessaire
=> assurer la testabilité d'une classe.

Utilisation de tests de haut niveau
=> permettent de déployer un filet de tests couvrant de larges portions de code legacy.

Lors d'un mélange de codes créer en TDD et de codes legacy, les programmeurs développent un excès de confiance lors des refactoring.

