TDD sur du code existant

Lionel DURAND

Avril 2023

lioneldurand@gmail.com
https://www.linkedin.com/in/lionel-durand-649b72193/

Code existant, code legacy ?

Du code reçu en héritage

- Que nous n'aurions probablement pas écrit / conçu ains
- Qu’il faut s'approprier

Ce code constitue une valeur

- Malgré les apparences
- La valeur doit absolument être préservée

Que fait ce code ? Comment le modifier sans risque ?

TDD & Legacy : comment

1. Ecrire des tests (de caractérisation)
2. Refactorer
3. Ajouter des fonctionnalités en TDD "classique"

Exercice : Trip service

Tests de caractérisation

Ecrire des tests sur un code legacy qui n'en a pas :

- pour appréhender son fonctionnel
- pour couvrir tout le code par des tests avant de le modif

Principes :

- Ne pas modifier du code non testé
- Commencer à tester "par la gauche"
- Vérifier la couverture

Tests de caractérisation : comment ?

- Appeler un morceau de code depuis un test
- Ecrire une assertion dont vous savez qu'elle échouera
- En échouant, le test indique quel est le comportement r
  du code
- Modifier le test de façon à ce qu'il attende le comporten
  que produit le code
- Répéter

Outils : IDE & Coverage

Refactorer

- Utiliser l'IDE !!!
- Petits pas / parallel change
- Refactorer par la droite (le plus spécifique d'abord)
- Isoler ce qui casse le flow d'exécution
- Réorganiser le code à l'intérieur des méthodes
    - Rapprocher les variables de leur utilisation
    - Travailler les if, les boucles,...
- Déplacer vers un objet le code/les méthodes qui le concernent

Refactorer

- Extraire des méthodes/variables/paramètres
- Inline (l'inverse de l'extraction)
- Casser les dépendances génantes
    - Par ex Singleton ou méthode statique
    - Extraire un paramètre
    - Extraire une méthode et surcharger/mocker
- Renommer, renommer, renommer !

Parallel Change

Technique pour l'ajout des changements non-retrocompatibles à une interface

- ajouter un nouvelle méthode mettant en oeuvre la nouv interface
- migrer progressivement les clients
- supprimer l'ancienne méthode

https://martinfowler.com/bliki/ParallelChange.html

Exercice : Tennis

Objectif

- développer en TDD une application qui compte les poin
  du tennis (un jeux uniquement)
- l'application doit retourner le score sur un jeux
- ex : "LOVE-LOVE", "FIFTEEN-LOVE", "DEUCE", "GAME PLA
  A",...

Contrainte : "nommage purement technique"

- un nom choisit au hasard ici
- pour une classe : 3 mots, une méthode : 2 mots, un
  param/une variable : 1 mot

Tennis : bilan

- Importance du nommage !
- Le code des tests est aussi important que le code métier
- Documenter un code legacy par les tests
- Encapsuler du code legacy si besoin (par ex via DSL dans tests)