Rappels sur le test (unitaire)

Mirabelle Nebut

janvier 2018
SVL - M1 Informatique

Outline

Qu'est-ce que tester ?

Définition historique : *[Myers] : Testing is the process of executing a program with the intent of finding errors*
Ce que vous avez commencé à voir en POO et COO :

- exécuter un morceau de code
- affirmer de manière exécutable le comportement attendu pour vérifier que le code fait ce qu'on veut

Ex : un test vérifiant que `ma_racine_carree(4)` vaut 2.
? Connaissez-vous d'autres types de test ?

Les quadrants Agile

http://istqbexamcertification.com/
what-are-test-pyramid-and-testing-quadrants-in-agile-testin

À chaque technique de test est associée un critère de test.

Exemples de techniques de test - SVL

Test unitaire : écrit par le développeur, exerce une toute petite partie du code testé dans l'objectif de vérifier qu'il fait bien ce qu'il veut

Test d'intégration : écrit par le développeur, exerce une partie plus grande de code dans l'objectif de vérifier que les différentes parties s'assemblent bien.

Test basé sur des critères de couverture : idem mais exerce le code dans l'objectif d'en couvrir la plus grande part possible.

(tous automatisés)

Exemples de techniques de test - IAGL

Test de performance : automatisé, exerce le logiciel de manière plus ou moins intense pour vérifier comment il supporte la charge.
Test d'acceptation : avec le client, automatisé ou manuel, permet de valider les fonctionnalités du logiciel (test système ou recette)
Test exploratoire : exploration manuelle du logiciel via ses interfaces pour y chercher des surprises résiduelles.
Test d'application web via leur interface utilisateur : automatisé, spécifique aux appli web.

Qui teste ?

Un testeur (spécialisé dans l'une ou l'autre des techniques de test), une fois qu'une fonctionnalité / le logiciel est terminé(e).
Le développeur, au fur et à mesure qu'il développe. SVL !

Pourquoi tester ?

Pour trouver des bugs, omniprésents dans les logiciels.
Au niveau unitaire : les tests servent de documentation interne exécutable.
Pour refactorer son code avec un filet de sécurité (test de non régression).
Pour améliorer la qualité de son code.
Pour s'éviter des veilles de release cauchemardesques.
Pour réduire les coûts de maintenance.
`https://medium.com/javascript-scene/the-outrageous-cost-of-skipping-tdd-code-reviews-57887064c4`

Pourquoi ne pas tester ?

Ça prend du temps d'écrire les tests (+30 à 50% et le temps, c'est de l'argent).
Ça double la quantité de code à maintenir.
Ça valide le modèle économique courant.
Mon boulot, c'est de coder, pas de tester.
Tu sous-entends que je code mal ?
J'utilise un langage fortement typé, si ça compile pas besoin de tests.

En SVL

Tests unitaires uniquement.
Techniques de tests en isolation : bibliothèque de mocks (? qui connaît ?)
Test-Driven Development : méthode, impact sur la conception (lien avec les patterns de COO)
Critères de test structurels : couverture de code
Test avancé : property-based testing

Anatomie d'un test unitaire

? À quoi ressemble un test unitaire ?

Anatomie d'un test unitaire

3 manières de se le rappeler :

- Arrange/Act/Assert
- Given When Then
- SetUp Exercise Verify

Ex: mot de taille max

Terminologie

Un test abstrait (sur papier, dans la tête) c'est :

- un objectif pour le test
- une donnée de test (valeurs pour les différents paramètres)
- le résultat attendu, inféré de la spécification.

L'ensemble du test exécutable s'appelle un cas de test.
La partie du test qui amène l'objet dans un contexte favorable est appelé fixture.
On regroupe les cas de test dans des suites de test.

Oracle

? Q'est-ce qu'un oracle ?

Oracle

L'oracle est toujours une assertion (avec plus ou moins de sucre syntaxique).
Pour le moment vous connaissez :

- les assertions booléennes
  - pour tester le retour d'une méthode
  - pour tester l'état d'un objet
- les assertions indiquant une levée d'exception

(plus tard) assertion attestant d'une interaction avec un objet

Absence d'oracle = smoke test

`assertTrue (true)` n'est pas un oracle digne de ce nom.
C'est la traduction de "si le test s'exécute en entier alors c'est que c'est bon".
Bien identifier l'oracle / le but du test.
Pas de tests pour le plaisir d'en avoir tout plein, après il faut les maintenir !

Résultat d'un test : verdict

? Quelles sont les issues possibles pour un test ?

Résultat d'un test : verdict

L'issue du test dépend de l'exécution de l'oracle :

- l'oracle est exécuté et indique que le test passe : succès
- l'oracle est exécuté et indique que le test échoue : échec
- une erreur se produit avant ou pendant l'oracle (levée d'exception) : erreur

Ex :

- `assertEquals(4,4)` : succès
- `assertEquals(4,3)` : échec
- `assertRaises(ValueError)` et levée de `ValueError` : succès
- `assertRaises(ValueError)` et aucune levée : échec
- `assertRaises(ValueError)` et levée de `SomeException` : erreur

Test unitaire / test d'intégration

Au sens strict, un test unitaire teste une unité de code (une classe dans notre cas) indépendamment de ses dépendances. On parle aussi de tests en isolation.
Dans la littérature, le terme est assez vague et désigne les tests écrits par les développeurs en utilisant un bibliothèque à la JUnit.
Mais si les dépendances sont instanciées, c'est déjà du test d'intégration.
Et c'est plus difficile de trouver d'où vient l'erreur.

Que teste-t-on ?

Approche utilisée en SVL : approche fonctionnelle.
On identifie pour la fonctionnalité / méthode testée ses sous-comportements :

- un objectif de test par sous-comportement
- cas nominaux + cas exceptionnels
- on choisit des données de test "en plein milieu" et "aux bornes"

Caractéristiques d'un bon test

- automatique : rien à faire à part regarder le verdict
- le nom est parlant et décrit ce qu'on souhaite tester. Des ex : `https://dzone.com/articles/7-popular-unit-test-naming`
- les tests sont indépendants les uns des autres (lancés dans un ordre qu'on ne maîtrise pas).
- court et lisible
- pour unittest et junit : pas de paramètres pour les méthodes de test

Compétences vues en CTD / TD cette semaine

Rappels de test unitaire :

- définition du test unitaire
- anatomie d'un test unitaire Python et JUnit4
- verdict : les possibles et leur signification
- les différents type d'oracle
- quoi tester
- caractéristiques d'un bon test
- le test comme documentation exécutable

Feedback

Requis à la fin de chaque CTD-TP.

- (+) ce qui marche bien
- (-) ce qui pourrait être amélioré

Pour que le feedback soit efficace:

- anonyme (ou pas)
- constructif