Validation & Vérification

Rapport de projet

MUTATION TESTING

SOUKPA Adou & ZBIRKA Christophe

M2 ILA 2017/2018

1)Introduction

Notre projet utilise la technique dite de « Mutation Testing ». Le but de cette technique est de vérifier la pertinence et la robustesse des tests unitaires d'un projet informatique. Elle consiste à réaliser des modifications à l'intérieur des fichiers bytecode d'un projet java maven. Après chacune de ces modifications, on réalise les tests unitaires et ont vérifie qu'ils détectent bien ces modifications. Ces mutations sont réalisées à l'aide de Javassist.

Notre projet réalise des modifications sur toutes les classes du projet maven cible. Pour chaque classe, toutes les méthodes sont passées en revue par les mutateurs (qui effectuent les modifications) qui réalisent sur celles-ci toutes les mutations possibles. Après chaque mutation, les tests sont effectués.

https://github.com/czbirka/ILA-VV-TP-MutationTesting.git

2) Solution

Notre projet peut réaliser quatre type de mutations :

- suppression de toutes les instructions du corps d'une méthode *void*.
- remplacement des instructions du corps d'une méthode *boolean* par les instructions « return true ; » puis « return false ; ».
- remplacement, dans toutes les méthodes, d'un opérateur arithmétique par un autre quand cela est possible.
- Remplacement, dans toutes les méthodes, d'un opérateur de comparaison par un autre quand cela est possible.

3) Fonctionnement

Le principe de fonctionnement de notre projet est le suivant :

Algorithme principal :

- chargement des données de configuration
- génération des mutateurs
- chargement des classes à modifier
- pour chaque classe, réalisations des mutations et des tests (voir algorithme suivant)
- création d'un fichier bilan avec tous les résultats

Algorithme pour chaque classe :

- chargement des méthodes de la classe
- pour chaque methode, vérification et modification par tous les mutateurs (voir algorithme suivant)

Algorithme pour chaque mutateur :

- si une mutation est possible (présence d'un opérateur arithmétique par exemple) :
  - sauvegarde de la méthode d'origine
  - réalisation de la mutation
  - réalisations des tests
  - sauvegarde de la mutation et des résultats des tests.
  - remplacement de la méthode modifiée par la méthode d'origine
- refaire ces opérations tant que c'est possible

4)Evaluation

Nous n'avons pas eu le temps d'évaluer correctement la rapidité et la fiabilité de notre projet.

5) Etat des lieux

Le projet fonctionne.

Il faut encore développer le nombre des mutations à réaliser par les mutateurs.

Il faut encore développer d'autres types de mutateurs. Par exemple, des mutateurs propres à des méthodes de type double ou int … ;

Il faut encore développer le rapport final qui, pour l'instant, ne contient pas de données statistiques sur les taux de réussite des tests ou sur les vitesses d'exécution ;

Point très important, il faut tester notre projet sur un projet maven de taille importante pour s'assurer que sa vitesse d'exécution est suffisamment rapide.

6)Conclusion

Le projet est fonctionnel mais demande encore beaucoup de travail pour finaliser tous les points signalés au chapitre précédent.
Il serait également intéressant de développer une interface Gui conviviale pour utiliser ce projet sur d'autres projets maven.