Assertion generation

Noureddine KADRI, Fahim MERZOUK
Février 2019

Le but de ce document est de fournir des informations concernant la génération des assertions, dans le cadre du module Software-Testing (VV).
Un cahier de charges détaillé nous a été fourni au début du projet.

1. Introduction:

Créer des assertions pour assurer tous les cas de tests possible est une tâche assez preneuse en temps et fastidieuse, ce qui implique que la cohérence des tests n'est pas toujours fiable.

Le but de ce projet est de créer un outil qui génère de nouvelles assertions dans le but de couvrir tous les cas de tests possibles.

2. Solution:

L'outil est un projet développé en Java, construit en Maven, un outil de construction de projets (Build).

Afin de renforcer les tests, l'outil prend en paramètre le chemin d'un projet Maven et produit en sortie des fichiers tests contenants les assertions existantes plus celles générées par l'outil.

Pour assurer la cohérence des assertions, l'outil exécute les cas de test et récupère la valeur des méthodes de champs public et de getter public, ensuite il vérifie si les valeurs résultantes sont les mêmes.

Nous avons utilisé Spoon, qui est une bibliothèque open-source qui permet de transformer et d'analyser le code source Java. Spoon fournit un métamodèle Java complet où tout élément de programme (classes, méthodes, champs, instructions, expressions...) peut être lu et modifié.

Pour tester notre outil, nous nous sommes basés sur JUnit, un framework open source pour le développement et l'exécution de tests unitaires automatisables. Le principal intérêt est de s'assurer que le code répond toujours aux besoins.

L'exécution de l'outil se fait par la commande suivante:
Java -jar [Jar File Path] [Maven Project Path]
Ou le premier paramètre est le chemin vers l'exécutable de l'outil, et le deuxième est le chemin vers le projet Maven dont l'utilisateur veut générer de nouvelles assertions.

3. Evaluation:

Nous avons créé de nouveaux projets Maven pour tester le bon fonctionnement de cet outil. Ce dernier parcourt l'ensemble des tests, ensuite fait la correspondance entre les champs des différentes classes et leurs getters public .

Si un getter n'est pas présent dans les méthodes de tests, l'outil observe les valeurs des méthodes, génère une nouvelle assertion contenant le getter et la valeur calculée.

En fin d'exécution, l'outil crée de nouveaux fichiers tests couvrants toutes les assertions.

Nos projets de tests étaient de différentes complexités et variants en nombre de packages utilisés.

4. Discussion:

Nous avons réalisé l'ensemble des spécifications demandés dans le cahier des charges.

Pendant le développement de cet outil, nous avons rencontré des difficultés pour manipuler le code source de nos classes de test. L'observation des valeurs des méthodes était largement compliquée mais nous avons pu la résoudre grâce aux bibliothèques fournies dans Spoon.

5. Conclusion:

Le projet réalisé dans le cadre de ce module est très intéressant et assez riche, nous avons amélioré nos compétences de conception de tests de validation, ainsi la manipulation du code source Java. C'était une expérience différente de qu'on a pu voir dans d'autres projets.