LOG3430 – Méthodes de tests et validation de logiciels

TP3- Test boîte blanche d'une application

Auteurs

Thierno Barry 1550237
Tomas Costanzo 1578677

Automne 2016
30 Octobre 2016

Introduction

Dans ce troisième laboratoire du cours de LOG3430(Tests et validation de logiciels), nous avons à analyser et implémenter des tests fonctionnels (boite blanche) du projet ListeChainnee implémenté au TP1 et testé avec la méthode de boite noire au TP2. Dans ce laboratoire, nous voulons obtenir une couverture de 100% sur toutes les branches. Nous avons utilisé la technique et la méthodologie de description utilisé dans le cours pour effectuer et présenter l'analyse.

Couverture pour la technique EC

La couverture de test EC par la méthode Black Box (boite noire) nous donne une couverture de 77% pour myListImpl, 68% pour ListeChaineeImpl et 88% pour setCalculatorImpl. Pour couvrir à 100% le code, nous allons implémenter une suite de test en boite blanche.

Test 1 : `union01_WhiteBox`

Ce test couvre la classe setCalculatorImpl. À la ligne 19, la branche else de la condition if(!setA.contains(a)) n'est pas couverte. Cette condition appartient à la méthode union.
Pour tester cette branche, nous ajoutons un nouveau test dans lequel il y a une valeur qui se répété dans les deux éléments qui sont passés à la méthode union. Le retour de la fonction setA.contains(a) va retourner une valeur true et comme !true = false alors la branche else sera couverte. Ce test n'aurait pas été découvert par la méthode boite noire car elle est spécifique à l'implémentation du code.

Test 2 : `removeAt01_WhiteBox`

Ce test couvre la méthode removeAt de la classe myListImpl. À la ligne 50, la branche de la condition if(pos == 0) n'est pas couverte. Il faut tester avec la méthode remoteAt(int pos) avec une position pos = 0 pour couvrir cette branche.

Test 3 : `removeAt02_WhiteBox`

Ce test couvre la méthode removeAt de la classe myListImpl. À la ligne 54, la branche à l'intérieur de la boucle while(pos-- > 1) n'est pas couverte car la valeur de pos n'a pas été testée avec une valeur plus grande que 1. Il faut tester avec la méthode remoteAt(int pos) avec une position pos = 2 pour couvrir cette branche.

Test 4 : `removeItem01_WhiteBox`

Ce test couvre la méthode removeItem de la classe myListImpl. Pour compléter la couverture, nous allons essayer d'enlever un item qui n'existe pas dans une liste vide.

Test 5 : `removeItem02_WhiteBox`

Ce test couvre la méthode removeItem de la classe myListImpl. Le code à partir la ligne 77 jusqu'à la ligne 81 n'est pas couvert par aucun test. Pour couvrir cette partie du code, il faut tester le cas dans lequel on essaie d'enlever un élément qui n'est pas au début de la liste.

Couverture pour la technique AC

La couverture de test AC par la méthode Black Box (boite noire) nous donne une couverture de 77% pour myListImpl, 68% pour ListeChaineeImpl et 88% pour setCalculatorImpl. Pour couvrir à 100% le code, les mêmes jeux de tests que pour la technique EC ont été implémentés et nous avons rajoutée une nouvelle méthode de test boite blanche :

Test 6 : `OperationParDefaut_WhiteBox`

Ce test couvre la classe ListChainneImpl. La condition par default du switch case n'était pas couverte. Le but de ce test est de passer une valeur invalide au switch pour lui forcer d'entrer dans la clause default a la ligne 73.

Couverture totale

La couverture des branches est de 100% sauf dans la classe ListeChaineeImpl qui donne une couverture de 68.4%. Nous attribuons ce manque de couverture a une erreur dans jococo. En effet, l'implémentation du code possède une condition multiple switch qui ne donne pas 100% de couverture lorsqu'on exécute les tests. Ce problème est attribué à la façon donc java gère les conditions switch mais en théorie la couverture du code devrait être de 100%.

Conclusion

Ce travail pratique nous a permis de nous familiariser et de mettre en pratique les notions acquises sur les tests fonctionnels (White box) dans le cadre du cours. Pendant ce laboratoire, nous avons appris à trouver et implémenter des cas de test spécifiques à l'implémentation du code et qui ne sont pas découverts avec la méthode de boite blanche.