LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 4

Tests OO - MaDUM

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Hiver 2022

1 Introduction

Dans ce travail pratique vous allez effectuer les jeux des tests orientés objets pour tester le module *crud.py* du système RENEGE.

2 Objectifs

Les objectifs généraux de ce laboratoire sont :

1. Pratiquer la conception des tests orientés objets, notamment avec la méthode MaDUM.

3 Mise en contexte théorique

Un logiciel OO possède des éléments spécifiques qui doivent être considérés lors des tests. Notamment, l'absence ou présence d'erreur n'est pas seulement représentée par une relation entrées-sortie, mais aussi par l'état interne de l'objet. Dans ce contexte, c'est important de faire les tests en considérant les séquences d'opérations. L'objectif est de trouver une séquence d'opérations qui mettra la classe dans un état contradictoire à ses invariants ou produira une sortie qui ne respecte pas son oracle.

Tester les classes pour toutes les séquences de méthodes souvent n'est pas possible. Dans ce travail pratique, vous allez utiliser la méthode MaDUM pour réduire le nombre de séquences à tester.

Approche MaDUM

MaDUM est une abréviation de Minimal Data Member Usage Matrix ou Matrice minimale d'utilisation des données membres d'une classe. Selon Bashir et Goel [1], les auteurs du livre Testing Object-Oriented Software : Life Cycle Solutions, le MaDUM se définit comme suit : « *A MaDUM is an* ***n*m*** *matrix, where* ***n*** *is the number of data members in the class and* ***m*** *represents the number of member functions in the class. An entry* $MaDUM_{i,j}$ *is marked for the correct usage type if the* j^{th} *member function manipulates its* i^{th} *data member in its implementation.* »

On utilise le MaDUM pour concevoir une stratégie de test. Bashir et Goel définissent aussi le terme de "tranche" ou "slice" en anglais, qui représente un quantum d'une classe avec un seul attribut et le sous-ensemble de méthodes pouvant le manipuler. La stratégie de Bashir et Goel est de tester une tranche à la fois et, pour chaque tranche, tester les séquences possibles des méthodes appartenant à cette tranche.

Avant d'identifier les "tranches", il faut catégoriser les fonctions membres de la classe à tester.

Il existe 4 catégories :

— Constructeurs (**C**) : constructeurs.

1. `https://bit.ly/3oQrE2W`

— Rapporteurs (**R**) : getters pour les données membres.
— Transformateurs (**T**) : setters ou toute autre méthode qui modifie l'état des données membres.
— Autres (**O**) : méthodes qui ne modifient pas l'état des données membres. Par exemple : affichage, destructeurs, etc...

L'étape suivant est de créer la matrice MaDUM, ou chaque ligne correspond à une "tranche", qui décrit l'utilisation d'attribut par les fonctions. Pour chaque tranche, il faut faire les tests suivants :

1. Tester les rapporteurs :
   — Modifier la valeur avec un transformateur et vérifier la sortie du rapporteur.
2. Tester les constructeurs :
   — S'assurer que les attributs sont correctement initialisés.
3. Tester les transformateurs :
   — Instancier l'objet sous test avec le constructeur ;
   — Créer toutes les séquences possibles de transformateurs (c.a.d 3 transformateurs - il faut tester $3! = 6$ séquences).
   — Si le transformateur contient des branches, tous les chemins où l'attribut qui est manipulé est modifié doivent être testés.
   — Vérifier la sortie des rapporteurs après les transformations.
4. Tester les autres :
   — Seule leur fonctionnalité comme une entité autonome a besoin d'être vérifiée.

Les tâches

La tâche, principale dans cette partie est de tester le module *crud.py* en utilisant l'approche MaDUM. Voici les étapes intermédiaires à suivre :

1. Concevoir une matrice MaDUM pour votre réalisation de la classe CRUD. La matrice doit avoir quatre attributs : "users_data", "groups_data", "users_lookup" et "groups_lookup".
2. Créer le fichier *test_crud_madum.py*. Dans le fichier, en utilisant l'outil Unittest python, réaliser les tests nécessaires selon l'approche MaDUM (C, R, T, O). Dans ce TP on vous demande de tester seulement la tranche qui correspond a l'attribut "groups_data". En tant que transformateurs il faut tester les fonctions "add_new_group", "update_groups", "remove_group", "remove_group_member", qui donne 24 combinaison possibles. Assurez-vous d'avoir une fonction test pour chaque séquence.
3. Important ! Avec les tests, il faut vérifier que si un groupe est supprimé, l'id unique va être assigné aux groupes ajoutés par la suite

4 Livrables attendus

Les livrables suivants sont attendus :
— Un rapport pour le laboratoire [8 points]. Dans le rapport :

1. Montrez la matrice MaDUM.
2. Indiquez les séquences d'appels faits pour chaque catégorie (C, R, T, O). Pour les transformateurs, limitez-vous aux 24 combinaisons. Indice : une séquence commence toujours par un constructeur et finit par un rapporteur.

— Le dossier contenant les modules crud.py et test_crud_madum.py, et les fichiers users.json, groups.json (si utilisés) [12 points]. Note : 2 points seront accordés si toutes les séquences présentes dans le rapport sont implémentées et que les tests passent. Nous vérifierons le code de 5 séquences choisies préalablement (2 points par test).

Mettre le tout dans une seule archive **zip** avec pour nom matricule1_matricule2_matricule3_lab4.zip à téléverser sur Moodle.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

Consultez le site Moodle du cours pour la date et l'heure limites de remise des fichiers. Un retard de]0,24h] sera pénalisé de 10%, de]24h, 48h] de 20% et de plus de 48h de 50%.