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

Automne 2021

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.
2. Vérifier la qualité des jeux des tests avec l'approche des mutations.

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 deux méthodes qui permettent de réduire le nombre de séquences à tester, notamment MaDUM et pré-/post- conditions.

Partie 1 : 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* $\boldsymbol{n}$**m matrix, where* $\boldsymbol{n}$ *is the number of data members in the class and* $\boldsymbol{m}$ *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 :

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

— Constructeurs (**C**) : constructeurs.
— 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 :
   — S'assurer qu'il y a transformateurs et rapporteurs pour chaque attribut .
   — Pour chaque attribut changer la valeur avec les transformateurs et assurer que la sortie du transformateur correspond à 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 doivent être testés.
   — Vérifier la bonne sortie des transformateurs avec les rapporteurs.
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. Designer un matrice MaDUM pour votre realisation du classe CRUD. Matrice doit avoir au moins deux attributs : "users_data" et "groups_data".
2. Créer le fichier *test_crud_madum.py*. Dans le fichier, en utilisant l'outil Unittest python réaliser les testes nécessaires selon l'approche MaDUM. Dans ce TP on vous demande de tester seulement la tranche qui correspond a l'attribut "users_data". En tant que transformateurs il faut tester les fonctions "add_new_user", "update_users", "remove_user", "remove_user_group", qui donne 24 combinaison possibles. Aussi, il ne faut pas implémenter les tests de type "autres".
3. Ajoutez le constructeur dans la classe CRUD (réalisation c'est a vous a décider.). Par exemple, dans le constructeur, vous pouvez initialiser les attributs "*users_data*" et "*groups_data*" avec un dictionnaire vide "{}".
4. Pour les fonctions transformateurs dans *crud.py* au lieu de la sortie booléen (True/False) il faut retourner la valeur de la variable qui était changée par la fonction.
5. Important ! Avec les testes, il faut assurer que :

— si l'utilisateur est supprimé, l'id unique va être assigné aux utilisateurs ajoutés prochainement ;
— Quand l'information pour l'utilisateur est modifiée, la date du dernier message vu va être changée si un message, qui vient de cet utilisateur, avec la date plus récente est détecté.

4 Livrables attendus

Les livrables suivants sont attendus :

— Un rapport pour le laboratoire [5 points]. Dans le rapport :
  1. Montrez les étapes pour créer le jeu de test pour approche MaDUM (en incluant la matrice MaDUM).
— Le dossier contenant les modules crud.py, test_crud_madum.py, et fichiers users.json, groups.json (si utilisés) [15 points].

Le tout à remettre dans une seule archive **zip** avec pour nom matricule1_matricule2_lab1.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%.