LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 3

Tests OO - MaDUM

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

Automne 2019

1 Mise en contexte théorique

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, 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, on teste les séquences possibles des méthodes appartenant à cette tranche.
Avant d'identifier les tranches de données et les séquences pour chacune, il faut catégoriser les fonctions membres de la classe à tester.
Il existe 4 catégories :

- Constructors (**C**) : constructeurs.
- Reporters (**R**) : getters pour les données membres.
- Transformers (**T**) : setters ou toute autre méthode qui modifie l'état des données membres.
- Others (**O**) : méthodes qui ne modifient pas l'état des données membres. Par exemple : affichage, destructeurs, etc...

Pour plus de détails et d'exemples, vous pouvez consultez les notes de cours sur les tests OO : https://moodle.polymtl.ca/pluginfile.php/522186/mod_resource/content/3/LOG3430_C07C_OO_Sequences.pdf

2 Objectifs

- Construire le MaDUM pour une classe sous tests.
- Identifier les tranches de données de la classe.
- Générer les séquences de test et les implémenter à l'aide de unittest.

3 Mise en contexte pratique

Pour réaliser ce travail, il faut commencer par construire le MaDUM en identifiant les données membres et les fonctions membres de la classe *Queue.py*
Ensuite, il faut identifier les tranches de données et les séquences pour chaque tranche.
Enfin,vous devez implémenter les séquences trouvées avec unittest.

4 Travail à effectuer

1. Construire le MaDUM en identifiant respectivement les constructors, reporters,transformers, et others pour les attributs de la classe *Queue.py*.

2. À l'aide de unittest, écrire une classe de test unitaire pour tester les tranches identifiées dans l'étape précédente. Pour chaque tranche, la séquence des méthodes doit suivre le principe de MaDUM.
3. Est-ce que cet example montre une ou plusieurs limitations de MaDUM ? Si oui, pouvez-vous les citer ?
4. À l'aide de l'outil Coverage.py, évaluez la couverture de la classe *Queue.py* et identifiez les parties de code non couvertes, s'il y en a. Pour les parties non couvertes, essayer de faire des tests boîte blanche pour atteindre la couverture maximale.

5 Questions

1. Qu'est ce qui se passe si on fixe le MAX de Queue à 2 ? Proposez une solution pour corriger le bug.
2. Quelle est la mauvaise pratique du codage qui a causé le bug ? Comment cette mauvaise pratique affecte le test MaDUM ? Proposez un refactoring du code pour éviter ce genre de bugs.

6 Livrables attendus

Les livrables suivants sont attendus :
— Un rapport pour le laboratoire.
— Le dossier COMPLET contenant le projet.

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%.