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



Soumis par

Roman Zhornytskiy (1899786)

Hakim Payman (1938609)

Gabriel Tagliabracci (1935775)

Groupe : 02

Soumis à Noureddine Kerzazi

Hiver 2020

4. Travail à effectuer

4.1. Identifier la situation la plus favorable pour un test MaDUM ensuite construire le MaDUM en identifiant respectivement les constructors, reporters, transformers, et autres pour les attributs de la classe huffman.py.

Le cas idéal pour un test MaDUM est un cas tel qu’il implique une quantité minimale de séquences d’opérations à exécuter pour réaliser les tests d’une tranche.

Tableau 1: MaDUM de la classe Huffman



4.2. 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.

d1 = <{Huffman(0, “1”) -> self.weight}, { self.weight == 0}>

d2 = <{from\_string(“testing”)}, { from\_string(“testing”).weight == len(“testing”)}>

d3 = <{Huffman(0, “1”) -> self.data}, { self.data == “1”}>

d4 = <{Huffman(0, “1”) -> self.build\_codebook()}, { self.data == “1”}>

d1 = <{Huffman(0, “1”) -> self.weight}, { self.weight == 0}>

**Roman : est-ce qu’on va faire ça ^^^^ pour tout le reste?**

4.3. Est-ce que cet exemple montre une ou plusieurs limitations de MaDUM ? Si oui, pouvez-vous les citer ?

Oui

4.4. À l’aide de l’outil Coverage.py, évaluez la couverture de la classe Huffman.py (dans le fichier huffman.py) et identifiez les parties de code non couvertes, s’il y en a. Pour les parties non couvertes, essayez de faire des tests boîte blanche pour atteindre la couverture maximale.

4.5. Qu’est-ce qui se passe si la fonction responsable du padding ne fonctionne pas comme attendu ? Proposez une solution pour corriger le bogue.

4.6. Complétez le cas de test suivant pour couvrir des cas extrêmes.

