INF3430 –Tests et validation

Automne 2020

TP No. 4

Groupe 1

1946801– Maude Desrosiers-Carbonneau

1946878 – Maxime Treamblay-Rheault

21 novembre 2020

Note pour l’exécution du programme

Puisque certaines fonctions étaient nécessaires sans être directement testées dans le contexte du TP, nous voulions éviter d’avoir des mutants à ces endroits. Nous avons donc utilisé une option spéciale --coverage pour l’exécution de mutpy.

Commandes à effectuer

mut.py --target crud.py --unit-test test_crud_madum.py -m --coverage

mut.py --target crud.py --unit-test test_crud_conditions_T.py -m --coverage

mut.py --target crud.py --unit-test test_crud_conditions_F.py -m --coverage

Partie 1: MADUM de la classe CRUD

	CRUD()	clean_files	add_new_user	add_new_group	read_users_file	remove_user_group
users_file	C	T	T	T (via appel à update_users)	R	T
groups_file	C	T	T (dans une branche)	T		T
user_counter	C	T	T			
group_counter	C	T	T (dans une branche)	T		
removed_users_ids	C	T				

	read_groups_file	get_user_data	get_group_data	get_user_id	get_group_id	remove_group
users_file		R		R		T
groups_file	R		R		R	T
user_counter						

group_counter						
removed_users_ids						

	modify_users_file	modify_groups_file	update_users	update_groups	remove_user	remove_group_member
users_file	T		T	T	T	T
groups_file		T		T	T	T(via appel à remove_user_group)
user_counter					T	
group_counter						
removed_users_ids					T	

Tranche MaDUM retenue

	read_users_file	CRUD()	add_new_user	remove_user_group	update_users	remove_user
users_file	R	C	T	T	T	T

Cas de test

Constructeur et rapporteur
d1 = <{CRUD(), read_users_file()}, {read_users_file={}}>

Ordre des Transformations après avoir effectué le constructeur

	T1	T2	T3	T4	résultat de read_user_file
d2	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	update_user("1", "name", "user1alt@email")	remove_user("1")	{}
d3	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	remove_user("1")	update_users("1", "name", "user1alt@email")	{}
d4	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	remove_user("1")	remove_user_group("1", "default")	{}

d5	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	remove_user("1")	{}
d6	add_new_user("user1@email", "2020-11-15")	remove_user("1")	remove_user_group("1", "default")	update_users("1", "name", "user1alt@email")	{}
d7	add_new_user("user1@email", "2020-11-15")	remove_user("1")	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	{}
d8	remove_user("1")	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec "Groups": ["default"]
d9	remove_user("1")	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	users_file contient user1@email avec "Groups": []
d10	remove_user("1")	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	update_users("1", "name", "user1alt@email")	users_file contient user1alt@email avec"Groups": []
d11	remove_user("1")	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	users_file contient user1alt@email avec"Groups": []
d12	remove_user("1")	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	users_file contient user1alt@email avec"Groups": ["default"]
d13	remove_user("1")	remove_user_group("1", "default")	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec"Groups": ["default"]
d14	update_users("1", "name", "user1alt@email")	remove_user("1")	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec"Groups": ["default"]
d15	update_users("1", "name", "user1alt@email")	remove_user("1")	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	users_file contient user1@email avec"Groups":[]
d16	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	remove_user_group("1", "default")	remove_user("1")	{}
d17	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	remove_user("1")	remove_user_group("1", "default")	{}

d18	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	remove_user("1")	{}
d19	update_users("1", "name", "user1alt@email")	remove_user_group("1", "default")	remove_user("1")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec"Groups": ["default"]
d20	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	remove_user("1")	{}
d21	remove_user_group("1", "default")	add_new_user("user1@email", "2020-11-15")	remove_user("1")	update_users("1", "name", "user1alt@email") }	{}
d22	remove_user_group("1", "default")	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	remove_user("1")	{}
d23	remove_user_group("1", "default")	update_users("1", "name", "user1alt@email")	remove_user("1")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec"Groups": ["default"]
d24	remove_user_group("1", "default")	remove_user("1")	add_new_user("user1@email", "2020-11-15")	update_users("1", "name", "user1alt@email")	users_file contient user1alt@email avec"Groups": ["default"]
d25	remove_user_group("1", "default")	remove_user("1")	update_users("1", "name", "user1alt@email")	add_new_user("user1@email", "2020-11-15")	users_file contient user1@email avec"Groups": ["default"]

Certaines variations de ces tests principaux ont étés ajoutés pour faire la couverture des branches.

Résultat final

[*] Mutation score [24.99885 s]: 90.4%
- all: 52
- killed: 45 (86.5%)
- survived: 5 (9.6%)
- incompetent: 0 (0.0%)
- timeout: 2 (3.8%)

[*] Coverage: 1085 of 1839 AST nodes (59.0%)

Analyse des résultats obtenus

```
  399:
  400:
  401:         del users[user_id]
  402:         self.modify_users_file(users)
- 403:         self.user_counter -= 1
+ 403:         self.user_counter += 1
  404:         self.removed_users_ids.append(user_id)
  405:
  406:         return True
  407:
```

Exemple de mutant vivant

L'exemple ci-dessus démontre un mutant non équivalent n'ayant pas été trouvé par le test MADUM. Il s'agit de la modification de l'incrémentation du user_counter dans remove_user(). Afin de découvrir ce défaut, il aurait fallu enchaîner plusieurs fois certains transformateurs et ceci ne fait pas partie de la méthode MADUM. En effet, la séquence add_user(), remove_user(), add_user() aurait permis de trouver ce mutant. Il y a 3 mutants faisant intervenir les compteurs qui auraient pu être découverts en appliquant la même méthode de répétition dans les séquences du MaDUM. Il y a également 2 mutants équivalents qui s'ajoutent au décompte pour un total de 5 mutants non trouvés.

```
   43:         month = int(date[5:7])
-  44:         day = int(date[8:10])
+  44:         day = int(date[8:])
   45:         if len(date) != 10:
   46:             return False
   47:
   48:
------------------------------------
[0.00779 s] survived
```

Exemple de mutant équivalent

Partie 2: approche pré- et post- conditions

Pré et post conditions

CRUD():

Pré: aucune

Post: users_file={}
groups_file = {}
user_counter = 1
removed_users_ids = []
group_counter = 1

add_new_user(user_email, date):

Pré: users_file != null and groups_file != null
and user_counter > 0 and group_counter > 0

Post: user_counter += 1
and user_email in user_file
and user_email in groups_file["1"] //Ajout de l'utilisateur au groupe par défaut
group_counter += 1

update_users(user_id, field, data):

Pré: user_id in users_file
and field in users_file[user_id]

Post: users_file[user_id][field] = data

get_user_data(user_id, field):

Pré: users_file != null
and user_id in users_file

Post: aucune

remove_user(user_id):

Pré: users_file != null
and user_id in users_file
and user_id > 0

Post: user_id in removed_users_ids
and users_file[user_id] = None
and user_counter -= 1
and L'utilisateur a été retiré des groupes auxquels il faisait partie

remove_user_group(user_id, group_name):

Pré: users_file != null and groups_file != null
and user_id in users_file
and group_name in users_file[user_id]["Groups"]
and group_name in groups_file

Post: group_name not in users_file[user_id]["Groups"]

Triplets:

CC1 = (CRUD, add_new_user, T) ,
CC2 = (CRUD, update_users, F),
CC3 = (CRUD, get_user_data, F) ,
CC4 = (CRUD, remove_user, F),
CC5 = (CRUD, remove_user_group, F)

CA1 = (add_new_user, add_new_user, T) ,
CA2 = (add_new_user, update_users, user_id in users_file),
CA3 = (add_new_user, get_user_data, user_id in users_file),
CA4 = (add_new_user, remove_user, user_id in users_file),
CA5 = (add_new_user, remove_user_group, user_id in users_file and group_name in users_file[user_id]["Groups"] and group_name in groups_file)

CU1 = (update_users, update_users, user_id in users_file and field in users_file[user_id])
CU2 = (update_users, add_new_user, user_id in users_file)
CU3 = (update_users, get_user_data, user_id in users_file and field in users_file[user_id])
CU4 = (update_users, remove_user, user_id in users_file)
CU5 = (update_users, remove_user_group, user_id in users_file and group_name in users_file[user_id]["Groups"] and group_name in groups_file)

CG1 = (get_user_data, get_user_data, user_id in users_file)
CG2 = (get_user_data, add_new_user, T)
CG3 = (get_user_data, update_users, user_id in users_file and field in users_file[user_id])
CG4 = (get_user_data, remove_user, user_id in users_file)
CG5 = (get_user_data, remove_user_group, user_id in users_file and group_name in users_file[user_id]["Groups"] and group_name in groups_file)

CR1 = (remove_user, remove_user, and user_id in users_file)
CR2 = (remove_user, add_new_user, T)
CR3 = (remove_user, update_users, user_id in users_file and field in users_file[user_id])
CR4 = (remove_user, get_user_data, user_id in users_file)
CR5 = (remove_user, remove_user_group, user_id in users_file and group_name in users_file[user_id]["Groups"] and group_name in groups_file)

CRG1 = (remove_user_group, remove_user_group, user_id in users_file and group_name in users_file[user_id]["Groups"] and group_name in groups_file)
CRG2 = (remove_user_group, add_new_user, T)
CRG3 = (remove_user_group, update_users, user_id in users_file and field in users_file[user_id])
CRG4 = (remove_user_group, get_user_data, user_id in users_file)

CRG5 = (remove_user_group, remove_user, and user_id in users_file)

Critère T pT

critère_T = {CC1, CA1, CU2, CG2, CR2, CRG2 }
citère_pT = {CA2, CA3, CA4, CA5, CU1, CU3, CU4, CU5, CG1, CG3, CG4, CG5, CR1 , CR3, CR4, CR5}
critere_TpT = citère_pT U critère_T

Séquences pour le critère T pT		
#Seq	Opération à effectuer	cas
séquence 1		
1	CRUD()	
2	add_new_user(user1@email, "2020-11-15")	CC1 → T
3	add_new_user(user2@email, "2020-11-15")	CA1 → T
4	update_users("2", "name", "newuser2@email")	CA2 → pT
5	get_user_data("1","name")	CU3 → pT
6	update_users("1", "name", "newuser1@email")	CG3 → pT
7	add_new_user(user3@email, "2020-11-15")	CU2 → pT
8	get_user_data("1","name")	CA3 → pT
9	get_user_data("1","name")	CG1 → pT
10	add_new_user(user4@email, "2020-11-15")	CG2 → T
11	remove_user("4")	CA4 → pT
12	add_new_user(user4@email, "2020-11-15")	CR2 → T
13	remove_user_group("4", "default")	CA5 → pT
14	add_new_user(user5@email, "2020-11-15")	CRG2 → T
15	remove_user("5")	transition CA4 → pT

16	remove_user("4")	CR1 → pT
17	update_users("1", "name", "newnewuser1@email")	CR3 → pT
18	remove_user("3")	transition CU4 → pT
19	get_user_data("1","name")	CR4 → pT
20	remove_user("2")	CG4 → pT
21	remove_user_group("1", "default")	CR5 → pT
séquence 2		
1	CRUD()	
2	add_new_user(user1@email, "2020-11-15")	CC1 → T
3	update_users("1", "name", "newuser1@email")	
4	update_users("1", "name", "newnewuser1@email")	CU1 → pT
5	remove_user("1")	CU4 → pT
séquence 3		
1	CRUD()	
2	add_new_user(user1@email, "2020-11-15")	CC1 → T
3	update_users("1", "name", "newuser1@email")	CA2 → T (déjà fait)
4	remove_user_group("1", "default")	CU5 → pT
séquence 4		
1	CRUD()	
2	add_new_user(user1@email, "2020-11-15")	CC1 → T
3	get_user_data("1","name")	CA3 (déjà fait)
4	remove_user_group("1", "default")	CG5 → pT
Séquences pour le critère F		

#Seq	Opération à effectuer	cas
1	CRUD()	
2	update_users("1", "name", "newuser1@email")	CC2 → F
Fin de cette séquence		
1	CRUD()	
2	get_user_data("1","name")	CC3 → F
Fin de cette séquence		
1	CRUD()	
2	remove_user("1")	CC4 → F
Fin de cette séquence		
1	CRUD()	
2	remove_user_group("1", "default")	CC5 → F

Résultat final pour le critère F

[*] Mutation score [2.86578 s]: 80.0%
- all: 10
- killed: 8 (80.0%)
- survived: 2 (20.0%)
- incompetent: 0 (0.0%)
- timeout: 0 (0.0%)

[*] Coverage: 258 of 1835 AST nodes (14.1%)

Analyse des résultats pour le critère F

Mutants ayant survécu:

```
    10:     def __init__(self):
    11:         self.users_file = dict()
    12:         self.groups_file = dict()
    13:         self.user_counter = len(self.read_users_file()) + 1
-   14:         self.group_counter = len(self.read_groups_file()) + 1
+   14:         self.group_counter = len(self.read_groups_file()) - 1
    15:         self.removed_users_ids = []
```

```
     9:
    10:     def __init__(self):
    11:         self.users_file = dict()
    12:         self.groups_file = dict()
 -  13:         self.user_counter = len(self.read_users_file()) + 1
 +  13:         self.user_counter = len(self.read_users_file()) - 1
    14:         self.group_counter = len(self.read_groups_file()) + 1
    15:         self.removed_users_ids = []
    16:
    17:
```

Dans les exemples ci-dessus, la modification de la valeur des counters n'est pas détectée par les tests, car les séquences pour le critère 'F' ne font pas intervenir des méthodes qui exploitent les "counters".

Résultat final pour le critère T pT

[*] Mutation score [11.63851 s]: 97.6%
- all: 41
- killed: 40 (97.6%)
- survived: 1 (2.4%)
- incompetent: 0 (0.0%)
- timeout: 0 (0.0%)

[*] Coverage: 934 of 1839 AST nodes (50.8%)

Analyse des résultats pour le critère T pT

Un seul mutant n'a pas été détecté. Dans ce cas, les compteurs pour les clés dans groups.json ne sont pas incrémentés correctement. Ceci n'est pas un mutant équivalent, mais n'a pas été détecté car son impact se fait ressentir au niveau des méthodes relatives au groupes tel que add_new_group(). Puisque ces méthodes ne sont pas testées ici, nous ne pouvons pas éliminer ce mutant.

```
--------------------------------------------------------------------------------
   55:                 'List_of_members': [user_email]}}
   56:
   57:
   58:             self.modify_groups_file(data)
-  59:             self.group_counter += 1
+  59:             self.group_counter -= 1
   60:         else:
   61:
   62:             groups = self.read_groups_file()
   63:             if user_email not in groups['1']['List_of_members']:
--------------------------------------------------------------------------------
[0.00525 s] survived
```

Comparaison entre les approches Tai Daniels et MaDUM

Points forts de Tai Daniels

L'approche Tai Daniels nécessite moins de cas de tests puisque la couverture des branches n'est pas requise et les combinaisons explorées se limitent à deux méthodes plutôt que quatre (comme MaDUM).

Les critères de test sont plus variés puisqu'on peut choisir entre T, F, T/pt, F/pF et une combinaison de celles-ci.

Si on utilise les 2 critères combinés de Tai Daniels, on arrive à tuer un plus grand nombre de mutants qu'avec Madum. En effet, ce nombre s'élève à 48 contre 45 dans le cas de MaDUM. Ceci pourrait s'expliquer par le fait que l'approche Tai Daniels, contrairement à MaDUM, permet de tester une même méthode deux fois consécutives. Ceci permet, entre autres, d'évaluer le comportement du système où deux utilisateurs sont ajoutés. Par exemple, cela permet de détecter des défauts éventuels relatifs au compteur 'user_counter' . Il est donc possible de tuer un grand nombre de mutants avec très peu de développement.

Il est aussi à noter que le mutation score de du critère T/pT est légèrement meilleur avec 97% contre 90% pour MaDUM.

Points forts de MaDUM

L'approche MaDUM offre une meilleure couverture à cause du nombre élevé de combinaisons de méthodes requises et de la couverture des branches. Notez que la couverture se situe à 59% alors que le critère F est à 14% et le critère T/pT est à 51%.

C'est en implémentant MaDUM qu'on a pu déceler le plus de problèmes liés à notre code puisque ça nous laissait voir des états incohérents de l'objet.