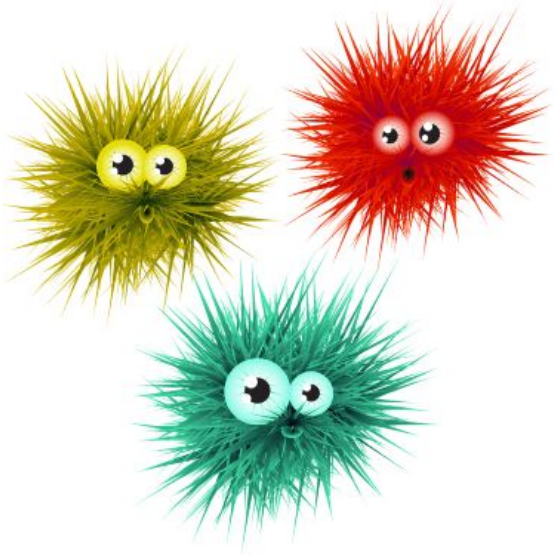
Rapport de V & V

Sujet 3 : Mutation Testing



Année 2017/2018	**Spécialité**
Master 2 Informatique	Ingénierie Logicielle en Alternance

BINÔME	◆ **Mhamed-Amine SOUMIAA** ◆ **Abdelghani MERZOUK**
UE	**V & V** **Validation et Vérification**

Introduction

Dans la pratique industrielle, les activités de vérification et de validation couvrent entre 35 et 55 % du coût de production de logiciel et cela sans tenir compte du coût de maintenance et d'évolution.

Le module de Validation et vérification (V&V) a pour but de former les étudiants à ce problème et de leur présenter les techniques de prévention des défauts (conception testable), de détection, localisation et correction des fautes (vérification et test).

Vérification → est-ce que le logiciel fonctionne correctement ?

Validation → est-ce que le logiciel fait ce que le client veut ?

L'objectif du projet était de choisir l'un des aspects des tests logiciel et puis l'implémenter en java sous forme d'un projet maven.

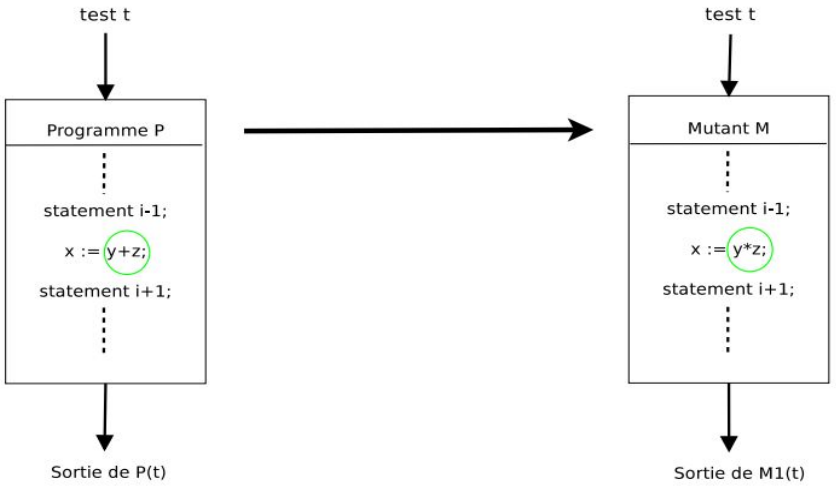
Mutation Testing

Les tests unitaires et d'intégration permettent de vérifier le fonctionnement d'une application et d'empêcher les régressions. Mais comment faire pour vérifier la qualité des tests ?

Une couverture du code par les tests à 100% ne signifie pas que le code est testé à 100%. Cette valeur indique seulement le pourcentage de code exécuté lors du passage des tests.

Une solution à cela : les tests par mutations (*mutation testing*). Ceux-ci permettent de vérifier la cohérence et la pertinence des tests unitaires.

Principe :



- Modifier le programme P en P' en injectant une modification syntaxique correcte (P' compile toujours)
- Idéalement, le comportement de P' est diff´erent de celui de P
- Sélectionner une donnée de test qui met en évidence cette différence de comportement (= tuer le mutant P')

Les tests par mutation dans notre projet

Dans notre projet on a choisit de procéder de cette façon :

Opérations choisies :

- Supprimer toutes les instructions dans la corps de la méthode `void`
- Remplacer le corps d'une méthode booléenne par une seule instruction `return true` (ou `return false`).
- Effectuez la substitution d'opérateurs suivante dans le contexte des expressions arithmétiques:

Operateur d'origine	Remplacé par
+	-
-	+
*	/
/	*

- Remplacer un opérateur de comparaison par un autre donne les substitutions possibles suivantes:

Operateur d'origine	Remplacé par
<	<=
>	>=
<=	<
>=	>

donc pour résumer :

Dans le projet /VV-Mutation-Testing-Testing/input

→ Création de nos classes arithmétiques

→ Test unitaire de chaque classe arithmétique

Dans VV-Mutation-Testing/Mutation-Testing

Les mutants implémentées sont:

-Opérateurs arithmétiques :

Opération + est remplacé par -

Opération - est remplacé par +

Opération * est remplacé par /

Opération / est remplacé par *

-Suppression du corps des méthodes de type void

-Renvoyer true pour les méthodes de types boolean

-Opérateur de comparaison :

Opération < est remplacé par <=

Opération > est remplacé par >=

Opération <= est remplacé par <

Opération >= est remplacé par >

Procédure suivie

1- Détecter tous les endroits que nous devons changer.$

2- Rendre la liste des emplacements trouvés.

->Pour chaque opérateur trouvé:

2- Remplacer l'opérateur dans le bytecode

3- Lancer la suite de tests

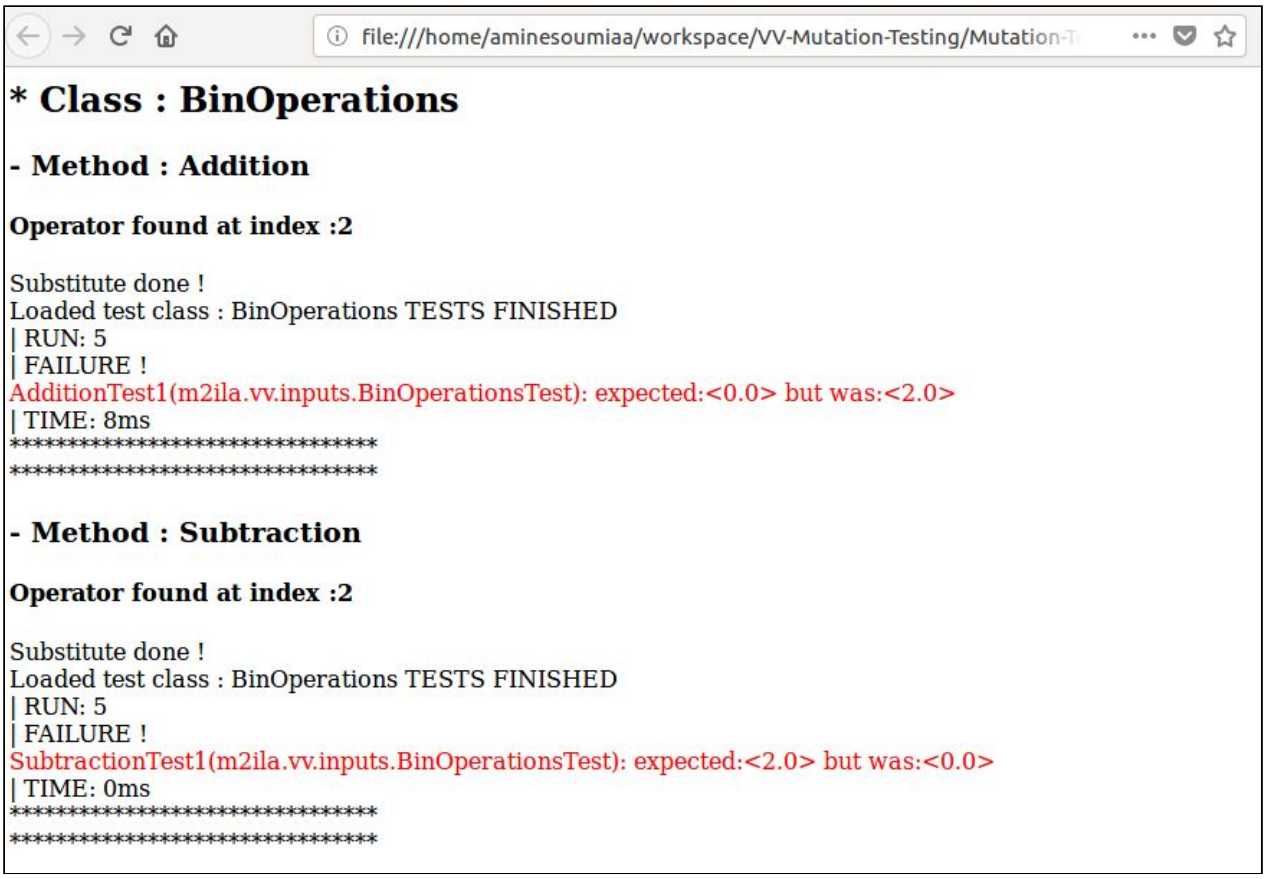
4- Insérer dans le rapport

5- Signaler si cela échoue

6- Annuler la dernière mutation

*Pour chaque mutation, nous créons une copie de la classe et exécutons des tests sur celle-ci.

Rapport généré



Lien github des sources du projet

https://github.com/masoumiaa/VV-Mutation-Testing

Conclusion

Ce projet qui intervient dans notre dernière année de Master Informatique a renforcé notre vision et notre compréhension des méthodes de tests et leurs fiabilité

Les tests de mutation sont un moyen simple et efficace de détecter la fiabilité des tests unitaires.

Ce projet de V&V a été pour nous une véritable occasion pour nous de voir de manière plus approfondie le monde de tests et plus particulièrement les tests par mutations (mutation testing) qui permettent de vérifier la cohérence et la pertinence des tests unitaires.