Tâche 1 - Métriques

Est-ce qu'il y a assez de tests?

1. TPC (tests par classe)
TPC peut servir à mesurer s'il y a assez de tests, car si TPC est grand, on considère que la majorité des fonctionnalités du code ont été testées. On peut utiliser TASSERT pour calculer le nombre de tests par classe. On veut un seuil raisonnable pour affirmer que la majorité des fonctionnalités ont été testées. JFreeChart est une grande bibliothèque pour la création de graphiques en Java. Alors, chaque classe supporte plusieurs fonctionnalités. Pour le seuil, on pourrait calculer le nombre de méthode moyen des classes de code multiplié par le nombre de tests par méthode moyen. Le seuil calculé est de 34.97.

2. PMNT (pourcentage de méthodes non testés)
On considère qu'il y a assez de tests si PMNT est petit. On compte NOM (nombre de méthodes) et le nombre de méthodes testées. On obtient PMNT par (NOM - le nombre de méthodes testées)/NOM * 100. Idéalement, toutes les méthodes seraient testées, mais si une méthode est très simple, des tests ne sont pas obligatoires. On décide alors que 80% des méthodes devraient être testées pour qu'il y ait assez de tests. Le seuil de décision est donc de 20%.

Est-ce que les tests sont à jour avec le reste du code?

1. AGE (age d'un fichier)
AGE fait référence au temps écoulé depuis la dernière modification. Si l'AGE du fichier test est plus petit que l'AGE du fichier contenant le code testé, alors on considère que les tests sont à jour. On peut calculer pour toutes les classes la différence suivante : AGE d'une classe - AGE de sa classe test. Puis calculer la moyenne des différences. Si la différence est positive, alors la classe test a été mise à jour après la classe cible. Si la différence est négative, alors la classe test n'a pas été mise à jour après que la classe cible a été mise à jour.

2. PMNT (pourcentage de méthodes non testés)
Si PMNT est petit, alors on considère que les tests sont à jour avec le reste du code. PMNT indique si les nouvelles méthodes ont été testées. On compte NOM (nombre de méthodes) et le nombre de méthodes testées. On obtient PMNT par (NOM - le nombre de méthodes testées)/NOM * 100. On garde le même seuil de décision posé pour la première question qui est de 20%.

Est-ce que les tests sont trop complexes?

1. Ratio taille code/taille test
Si le ratio est très faible, alors les tests sont trop complexes. Pour cette métrique, on va considérer qu'un seuil assez bas voudrait dire que les tests sont complexes par rapport au code. Le seuil de décision pourrait être de 1. Un ratio plus petit que 1 signifie que la taille des tests est plus grande que la taille du code, ce qui indique clairement que les tests sont trop complexes. On peut utiliser TLOC du Tp1 pour calculer la taille du code et celle des tests.

2. CC (complexité cyclomatique d'une méthode)
Si la complexité cyclomatique d'une méthode de test est élevée cela voudrait dire que les tests sont difficiles à comprendre, et donc on a une grande complexité. On pourrait calculer une moyenne de complexité en calculant la complexité de chaque test et en la divisant par le nombre de tests. Un seuil de décision serait de 3. Si la moyenne est inférieure à 3, cela voudra dire que les tests ne sont pas trop complexes. Dans le cas contraire, ils sont trop complexes.

Est-ce que les tests sont suffisamment documentés?

1. DC (Densité de commentaire)
Les commentaires sont utilisés pour expliquer le code et les tests, ils font partie de la documentation. Une grande densité indiquerait une documentation suffisante. Pour mesurer DC, on calcule CLOC/LOC. CLOC = LOC - TLOC du TP1 et on peut avoir LOC en comptant toutes les lignes en ignorant les lignes vides seulement. On pourrait considérer qu'une densité de commentaire de 30% indiquerait que les tests sont suffisamment documentés. Le seuil serait alors de 0.3. Si la moyenne des densités est supérieure à 0.3, cela voudrait dire que les tests sont assez documentés.

2. Ratio complexité/densité
Si une classe test contient des tests complexes, cette classe test devrait contenir plus de commentaires pour expliquer les tests. Si une classe test contient des tests peu complexes, cette classe test n'a pas besoin de plusieurs commentaires pour expliquer les tests. Pour mesurer cette métrique, on utiliserait une métrique pour calculer la complexité et la densité de commentaire. Nous avons décidé que le seuil de décision de la complexité cyclomatique est de 3 et celle de la densité est de 0.3, alors 3/0.3 = 10 serait notre seuil. Si le ratio est plus petit que 10, cela signifie que les tests de la classe sont assez documentés. Si le ratio est plus grand que 10, cela signifie que les tests ne sont pas assez documentés.

Tâche 2
Nous avons décidé de mesurer nos métriques de manière globale. Nous avons donc écrit notre code de manière à ce qu'il mesure nos métriques pour l'ensemble du projet JFreeChart.

Pour le nombre de tests par classe (TPC), on additionne le nombre de tests pour toutes les classes et on divise par le nombre de classes pour obtenir une moyenne de tests par classe.

Pour le pourcentage de méthodes non testées, nous avons calculé le nombre de méthodes de chaque classe et le nombre de méthodes testées. Nous avons présumé que pour tester une méthode de la classe cible, une méthode dans la classe test est créée. On obtient alors PMNT par (NOM - le nombre de méthodes testées)/NOM * 100.

Pour l'âge, nous avons utilisé la méthode `lastmodified()` pour obtenir le temps depuis la dernière modification. `lastmodified()` donne ce temps en millisecondes, nous l'avons changé pour des jours avec `TimeUnit.MILLISECONDS.toDays(fichier.lastModified())`. Nous avons calculé la somme des différences entre l'âge d'une classe et l'âge de sa classe test, puis nous avons obtenu la moyenne en divisant cette somme par le nombre de différences additionnées.

Pour le ratio taille code/taille test, nous avons utilisé TLOC de notre TP1 pour calculer le nombre de lignes dans les classes. Nous avons fait la somme de toutes les classes dans le dossier main divisé par la somme de toutes les classes dans le dossier test.

Pour la complexité cyclomatique, nous avons implémenté tout d'abord une méthode pour calculer la complexité cyclomatique de chaque méthode dans une classe java et retourner cette somme divisé par le nombre de méthodes de la classe. La méthode implémentée retourne donc la complexité cyclomatique moyenne d'une classe. Puis, nous avons appliqué cette méthode pour obtenir la somme des complexités cyclomatiques moyennes de toutes les classes test de JFreeChart divisée par le nombre de classes tests. Nous avons donc comme résultat la moyenne des complexités cyclomatiques de toutes les classes test du projet.

Pour la densité, nous avons implémenté tout d'abord une méthode pour calculer LOC, puis une méthode pour calculer la densité avec DC = CLOC/LOC, CLOC étant égal à LOC - TLOC que nous avions implémenté dans le TP1. Enfin, nous avons la moyenne des densités des classes test.

Pour le ratio complexité/densité, nous utilisons les méthodes implémentées pour la densité et la complexité cyclomatique. On calcule la somme des ratios de toutes les classes tests divisée par le nombre de classes test. On obtient donc la moyenne du ratio complexité/densité des classes test de JFreeChart.

Nous avons obtenu les résultats suivants :
Moyenne des TESTS PAR CLASSE = 26.35
Moyenne des différences d'AGE (AGE classe code - AGE classe test) = -1116
RATIO TAILLE CODE/TAILLE TEST = 2.329369509294763
Moyenne des COMPLEXITÉ CYCLOMATIQUE = 1.0345557429039385
POURCENTAGE DE MÉTHODES NON TESTÉES = 60.32236400268637
Moyenne des DENSITÉ DE COMMENTAIRE = 0.4446262993320302
Moyenne du RATIO COMPLEXITÉ/DENSITÉ = 2.5484962739137376

Tâche 3

Est-ce qu'il y a assez de tests?
La moyenne de tests par classe est de 26.35 ce qui est plus petit que le seuil choisi. Selon cette métrique, il n'y a alors pas assez de tests.

Le pourcentage de méthodes non-testées dans JFreechart est de 60.32%. C'est un grand pourcentage, car nous avons posé notre seuil à 20%. On en conclut qu'il n'y a pas assez de tests.

Les deux métriques choisies indiquent qu'il n'y a pas assez de tests.

Est-ce que les tests sont à jour avec le reste du code?
La moyenne des différences d'AGE est de -1116 jours. La moyenne est négative, cela indique que les tests ne sont pas à jour.

Le pourcentage de méthodes non-testés dans JFreechart est de 60.32%. C'est un grand pourcentage, car nous avons posé notre seuil à 20%. On en conclut que les tests ne sont pas à jour.

Les deux métriques choisies indiquent que les tests ne sont pas à jour avec le reste du code.

Est-ce que les tests sont trop complexes?
Le ratio taille code/taille test est de 2.33, ce qui signifie que la taille du code est environ deux fois plus grande que la taille des tests. Donc, selon la métrique ratio taille code/taille test, les tests ne sont pas trop complexes.

La moyenne des complexités cyclomatiques des classes tests de JFreeChart est de 1.03. Ainsi, cette métrique indique que les tests ne sont pas trop complexes, puisque la complexité cyclomatique est en dessous de notre seuil qui est de 3.

Les deux métriques choisies indiquent que les tests ne sont pas trop complexes.

Est-ce que les tests sont suffisamment documentés?
La moyenne des densités de commentaire des classes test de JFreeChart est de 0.4446. Puisque le seuil choisi est de 0.3, nous en concluons que les tests sont suffisamment documentés.

Le ratio complexité/densité obtenu est de 2.55 ce qui est bien plus petit que 10, le seuil choisi pour cette métrique. Ainsi, cela signifie que les tests sont suffisamment documentés.

Les deux métriques choisies indiquent que les tests sont suffisamment documentés.