Tabta Abderrahim et Berrachedi Wassim
https://github.com/leaderbear/IFT3913-TP2

Tâche 1

Question 1

Le niveau de documentation des classes est-il approprié par rapport à leur complexité ? Autrement dit, on s'intéresse ici à la facilité d'analyse. Pour répondre à cette question, nous proposons les deux métriques suivantes : La moyenne de la densité de commentaires et le nombre de commentaires par méthode dans une classe.

D'abord, on peut tous s'entendre que afin de documenter et d'expliquer son code, il est nécessaire d'avoir des commentaires. Pas assez ou trop de commentaires peut rendre compréhension assez complexe. Une densité de commentaire équilibrée peut être un indicateur d'une bonne documentation. Selon l'étude de Dr. Dirk Reihle et Olivier Arafat[1], la moyenne de densité de commentaire par projet est de 19%. Dans notre cas, nous sommes peu sévères et nous allons considérer qu'une densité entre 15% et 40% est correcte.

Pour prendre en compte le paramètre de la complexité, nous allons aussi considérer le ratio entre le nombre de fonctions et le nombre de commentaires par classe. Selon nous, le nombre de fonctions est indicateur de la complexité. Plus de fonctions nécessite plus de commentaires. Encore une fois, on cherche un seuil entre 0,15 et 0,40. Ainsi, s'il y'a beaucoup de méthodes et pas assez de commentaires on se retrouve rapidement en bas de 0.15. Pour répondre oui à cette question, les deux métriques doivent être respectées.

Question 2

La conception est-elle bien modulaire ? Nous parlons ici de la facilité de modification. Nous avons choisi les métriques CBO et LCOM. Car selon nous, lorsqu'il s'agit de mesurer la modularité de la conception, il est très important de considérer le couplage et la cohésion. En effet, nous avons bien retenu de notre cours de génie logiciel et une autre recherche[2], qu'il faut minimiser le couplage et maximiser la cohésion.

Selon une autre recherche[3], pour minimiser le couplage, on vise un cbo entre 1 et 4. Nous avons décidé d'être plus raisonnable puisque nous allons considérer la moyenne. Nous allons donc considérer un cbo entre 1 et 6.75 comme étant bon. Ensuite, pour LCOM, nous avons décidé d'utiliser une version normalisée. Lorsque LCOM est égale à 0, nous avons une cohésion parfaite, tandis que lorsque LCOM est égale à 1, nous avons une cohésion nulle. Puisque nous considérons la moyenne, nous ne voulons pas être sévère et nous avons donc considérer LCOM inférieur à 0.6 comme était suffisant. Encore une fois, pour répondre oui à cette question, les deux conditions doivent être respectées.

Question 3

Le code est-il mature ? Nous parlons ici de la stabilité. Nous allons considérer la dernière mise-à-jour du projet et le pourcentage de méthodes non testés. Notre raisonnement est le suivant : Un projet qui n'a pas été mis-à-jour depuis longtemps est obsolète et donc immature. De plus, un code qui est mis-à-jour, mais qui n'est pas bien testé est aussi immature. Donc, le code doit être maintenu et bien testé. Pour la première métrique, nous allons simplement calculer le nombre de mois depuis la dernière mise-à-jour des

[1] https://dirkriehle.com/wp-content/uploads/2009/02/icse-2009-nier-for-web.pdf
[2] https://www.maxwell.vrac.puc-rio.br/29079/29079_3.PDF
[3] https://objectscriptquality.com/docs/metrics/coupling-between-object-classes-cbo#:~:text=Analysis,the%20class%20is%20loosely%20coupled,

fichier java. Nous allons accorder une note sur 100 où chaque mois de plus sans mise-à-jour va pénaliser par 5 points de moins. Pour être plus clair, note = 100 – 5 * mois. Si le projet a été mis-à-jour y'a deux mois, la note sera de 90%. Une note supérieure à 70% est suffisante. Et pour la 2ème métrique nous allons tout simplement calculer le pourcentage de méthodes non testées et on s'attend à ce qu'au moins la moitié (50%) des méthodes soient testées. Pour répondre oui à cette question, les deux conditions doivent être respectées.

Question 4

Le code peut-il être testé bien automatiquement ? Ici, on va utiliser encore une fois le cbo. Un code avec un mauvais couplage peut s'avérer très difficile à tester puisque toutes les classes seront liées entre elles. Nous allons donc utiliser exactement le même seuil qu'à la question 2. Ensuite, nous allons considérer le temps de tests. En effet, selon nous, il est difficile de tester automatiquement si les tests prennent beaucoup de temps et de ressources. Donc, nous allons calculer le temps total nécessaire pour faire tous les tests du projet. S'il est inférieur à 5 secondes, alors on considère que le code est facile à tester. Les deux conditions doivent être respectées pour répondre oui à cette question.

Tâche 2

Nous procédons en deux étapes, d'abord nous collectons toutes les données brutes des métriques présentées précédemment. Ensuite, nous analysons ces données pour savoir si nos seuils sont respectés. Pour la première étape, nous avons utilisé 2 outils externes et un outil fait par nous-même. Chaque outil donne environ 2-3 métriques. D'abord, nous avons utilisé l'outil ck. Celui-ci, prend en entrée le dossier d'un projet et calcule le cbo, lcom et le nombre de méthode dans chaque classe pour retourner un fichier csv. Nous avons utilisé JUnit4 inclus dans Intelji qui calcule nombre de méthodes testées ainsi que le temps de tests pour produire un fichier xml ou html. Enfin, notre outil inspiré du tp1 calcule la densité de commentaire et la date de chaque fichier pour retourner un fichier csv. Il suffit ensuite, de lire et d'analyser ces fichiers pour chercher les données que nous avons besoin. Quand nécessaire on fait quelques calculs comme la moyenne. Voir le readMe pour plus de détails sur le fonctionnement de l'analyse. Enfin, on crée un fichier dans lequel on écrit si les seuils sont respectés.

Tâche 3

Notre analyseur répond automatiquement aux questions. Pour répondre oui à une question, les seuils des deux métriques doivent être respectés. Voici le résultat sur jfree (Résultats des métriques\result.txt) :

Q1 : Moyenne densité de commentaire : 0.15 < **0.344** < 0.4 ✓
Moyenne commentaire par fonctions 0.15 < **0.306** < 0.4 ✓
Q2 : Moyenne CBO 0.0 < **6.5** <6.75 | Moyenne LCOM 0.42 < 0.6 ✓
Q3 : Note dernière mise-à-jour **80%** > 70% | PMNT **37,2**% < 50% ✓
Q4 : Durée de test : **2425**ms < 5000ms | Moyenne CBO 0.0 < **6.5** <6.75 ✓

Nous avons répondu oui aux quatre questions, car tous les seuils fixés sont respectés. Nous avons obtenu ces résultats avec notre programme qui analyse les données des métriques. Puisque nous n'avions pas été très sévères, nos métriques peuvent dire lorsqu'un programme est mauvais, mais pas différencier un programme moyen et bon. Nous concluons donc que jfree est au moins de qualité moyenne.