IFT 3913 TP2

21 octobre 2022
Hugo Carrier 20197563
Maggie Robert 20182443

Métriques choisies pour les questions :

Q1 : Nous avons choisi de mesurer la densité de commentaires d'une classe comme première métrique. Si le niveau de documentation augmente avec la complexité alors on attend une corrélation positive entre CLOC et LOC. Une ligne sur 3 à une ligne sur 2 de commentaire est une bonne densité. Comme deuxième métrique nous avons utilisé le pourcentage de classes non testées. Les tests sont un type de documentation « exécutable », alors on attend un pourcentage bas, surtout s'il y a beaucoup de méthodes (et par conséquent beaucoup de complexité). On souhaite se rapproche le plus de 0 classe non testée.

*Q2 : P*our répondre à la deuxième question, nous avons choisi comme métrique de mesurer CSEC et la taille physique des classes dans le répertoire main. Si la conception est modulaire on attend à des valeurs base pour le CSEC. En considérant des classes utiles, les interface et les classes abstrait, environs un couplage de 10 serait raisonnable. Une petite taille physique est attendue, car cela indiquerait que chaque classe a une fonctionnalité spécifique, donc bien séparée en module.

Q3 : Pour évaluer la maturité du code nous avons utilisé comme métriques le nombre de jours depuis les commit et le nombre de lignes modifiées de ces commits. Nous avons utilisé les vingt derniers commits sur le code et le nombre de lignes ajoutées et supprimées de ceux-ci, pour avoir une bonne visualisation de l'état actuel. Si le nombre de jours (moyenne ou médiane) est très grand, soit plus de 100, cela nous indique que le projet est mature car il n'y pas beaucoup de travail à faire. Le nombre de lignes modifiées vient confirmer le tous. Lorsque ce nombre est grand, cela veut dire que le code est en construction. Peu de ligne de codes indique souvent que seulement des bogues ont été réglés, synonymes souvent de support pour un projet mature. Des modifications de moins 50 lignes semble très raisonnable. 50 lignes peuvent être un changement de 2 ou 3 classes, donc une correction de quelques choses.

Q4 : Nous avons choisi de mesurer le ratio taille code sur taille test. Plus cette valeur est basse, plus le code est bien testé. Selon l'opinion de d'autres développeur sur stackoverflow, un ratio de 3 est une valeur qui revient souvent dans des projets bien testé. Comme deuxième métrique on a choisi le pourcentage de classes non testées. Si le code est bien testé, ce pourcentage devrait être bas car on s'attend à ce que la plupart des classes publiques aient des tests. On veut être proche de 0 le plus possible.

Procédure de mesure :
Tous les mesures ont été implémenter par nous-même (https://github.com/3Pi1416/IFT3913-A-A22-TP2) . Durant la prise de mesure, le code prend souvent comme entrée le dossier du code source et

quelques fois le dossier du code source et le dossier des tests. Chaque métrique évalue chacun des fichiers java à l'intérieur. Tous les fichiers non-java ne sont pas pris en compte.

Réponse aux questions :
Q1 : La densité de commentaires maximale et environ 0.93, ce qui vaut dire qu'il existe une classe avec beaucoup de commentaires et très peu de code, mais le graphique nous montre que celle-ci se retrouve dans une classe avec peu de code, donc une donnée aberrante, surtout en regardant la régression en annexe. La médiane de la densité est de 0.6 ce qui nous indique que le code est très bien commenté. Seulement 30% des classes du projet jfreechart ne sont pas testées. Grâce à ces constations, nous pensons que le niveau de documentation est approprié pour la complexité.

Q2 : Les valeurs min et max pour CSEC sont 0 et 267 respectivement. Cependant, la médiane est 9 et la moyenne est environ 16. Ceci indique que même s'il y a des classes avec beaucoup de couplage, la plupart des classes ont un score CSEC assez petit. On voit la même tendance avec la taille physique des fichiers; la plupart des fichiers ont une taille raisonnablement petite (environ 5.6 kB) avec quelques-uns avec une taille beaucoup plus large. Alors en fonction de ces données collectées nous pouvons constater que oui la conception est bien modulaire sauf quelques cas exceptionnels.

Q3 : Nous avons comme résultats une médiane de 286 jours entre le moment où nous avons roulé le code et le moment des commits. Nous semblons avoir un code mature car il y a eu peu de code fait durant les derniers jours. De plus, le nombre médian de ligne supprimé est 12.5 et le nombre de lignes ajouté est de 6.5, donc la plupart de ces commits était très petit, indiquant une fois de plus un signe de maturité du projet. En conclusion, le projet serait mature.

Q4 : Le code semble pourvoir être en grande partie automatiquement testé. Nous avons 31% pour le pourcentage des classes non testées si nous excluons les interfaces, les classes privées et les autres types de fichier java qui ne sont pas des classes. Cette valeur n'est pas catastrophique mais plus proche de 0 aurait été mieux. Pour accompagner cette métrique, nous avions regardé le ratio de taille de code sur taille de test. Elle est un proxy qui nous indiquer si les tests englobent beaucoup de cas. Nous avons comme valeur 2.8 qui indique que la taille des tests est raisonnable. Il est donc possible de répondre que la plupart du code peut bien être automatiquement testé, mais il y a place à amélioration.

Niveau de maintenabilité de jfreechart :
En général le projet jfreechart est assez maintenable. Le facteur le plus indiquant est le niveau de maturité. Il y a encore des petits ajustements qui s'effectuent périodiquement, mais sans trop de changement majeur. Ceci nous assure que la fonctionnalité générale ne devrait pas changer. On a vu aussi que le code est modulaire. Ceci indique un logiciel maintenable, surtout pour des langages de programmation orientés objet. Le niveau de couverture des tests est également décent. Cela indique encore une fois que le projet est maintenable, car tout changement doit s'assurer que les tests existants passent toujours, protégeant ainsi la fonctionnalité du logiciel. Même s'il n'y a pas une augmentation de la documentation par rapport à la complexité, le projet reste quand même documenté. Tous ces facteurs mènent à un logiciel maintenable.

Annexe :
Graphique de densité de commentaires: