Alexis Boucher 20217120 Marguerite Mireille Camara 20143122 Lien GitHub:

https://github.com/AlexisBoucher/IFT3913

Tâche 1: Plan GQM

Goal:

Analyser la dernière version stable du code de la branche master du JFreeChart pour évaluer sa facilité d'analyse du point de vue du chef du projet.

Question:

Q1 : Est-ce qu'il y a assez de tests?

Q2 : Est-ce que les tests sont à jour avec le reste du code ?

Q3: Est-ce que les tests sont trop complexes?

Q4 : Est-ce que les tests sont suffisamment documentés ?

Métriques :

Question 1:

TPC (tests par classe):

TPC nous permettra de calculer le nombre de tests dans chaque fichier test, elle nous sera utile pour évaluer la quantité des tests effectué. Si TPC est relativement faible par rapport à la taille du projet, cela peut indiquer des lacunes dans les tests, ce qui signifie qu'il pourrait y avoir des parties du code non couvertes par les tests. Pour le mesurer nous modifierons le programme tls que nous avons implémenté lors du tp1 pour nous donner en plus le tassert moyen, ce qui est équivalent à Tpc.

TPP (tests par package):

Nous allons collecter le nombre de test présent dans chaque package que nous, avons, nous évaluerons la moyenne de tous les TPPs. Un seuil pour le nombre de test par package est fixé sur la base de la complexité du projet, une moyenne des TPPs en dessous du seuil sera un indice que le projet n'est pas suffisamment testé. Pour le mesurer nous modifierons le programme tls que nous avons implémenté lors du tp1.

PCC (Pourcentage code couvert):

Mesure le pourcentage de ligne de code non testées dans chaque package. Si un package a un pourcentage élevé de ligne de code non testées, cela pourrait indiquer que ce package est insuffisamment testées. Un seuil sera fixé, si le PCC est en dessous de ce seuil cela signifiera que le package concerné a besoin d'amélioration sur la couverture de ses tests. Pour le mesurer nous utiliseront le plug-in pour projet maven jacoco.

Question2:

AGE (âge d'un fichier):

Nous permettra de déterminer si nos fichiers tests sont plus vieux que les fichiers des classes qu'ils testent. Un seuil d'un certain nombre de jour est fixé, si la moyenne des fichiers tests sont en dessous du seuil cela signifiera que la majorité des tests sont plus vieux que les fichiers qu'ils tests, et donc ne seront pas considéré à jour. Nous ferons une implémentation de age pour mesurer celui-ci.

NCH:

Calcule le nombre de commits dans l'historique d'une classe, pour chaque classe de test cette metrique compte le nombre de commits associés à cette classe depuis le début de son historique. Nous comparerons les valeurs de NCH des classes de tests à celles du reste du code. Si le NCH des classes de tests est significativement inférieur à celui du code source principal, cela peut indiquer que les tests ne

https://github.com/AlexisBoucher/IFT3913

sont pas mis à jour aussi fréquemment que le code source, ce qui pourrait indiquer que le code n'est pas à jour.

PCC (Pourcentage code couvert):

Mesure le pourcentage de ligne de code non-testé. Un pourcentage élevé nous indiquera que beaucoup de code n'est pas testé, ce qui pourrait suggérer un ajout de lignes de code sans ajout de tests pour les vérifiés, ce qui peut nous indiquer que les tests ne sont plus à jours

Question3:

Ratio taille code / taille test :

Mesure la taille du code de test en comptant le nombre total de lignes de code dans les classes tests. On mesurera la taille du code source en comptant le nombre total de lignes de code dans le code que nous testons. On fera ensuite le ration Ratio taille code / taille test. Des seuils appropriés en fonction de JFreeChart sont fixés. Un ratio faible peut indiquer que les tests sont proportionnellement plus complexes que le code de la classe testée. Un ratio élevé, indiquera que les tests sont relativement plus simples que le code source. Nous utiliserons l'application cloc pour le mesurer.

NCLOC:

Calcule le nombre de lignes de code non-vides qui ne sont pas des commentaires. Un NCLOC élevé dans les fichiers tests par rapport à la taille de JFreeChart pourrait indiquer que les tests sont trop complexes. Nous utiliserons l'application cloc pour le mesurer.

TCMP:

Mesure ratio entre le nombre de ligne de code et le nombre de tests dans une classe (TLOC/TASSERT). Nous évaluerons le TCMP moyen de toutes les classe de test de jfreechart. Un ratio élevé nous indiquera qu'en moyenne les tests sont trop complexe et un ratio faible nous indiquera que les tests ne sont pas trop complexes. Nous modifierons le programme tls pour obtenir TCMP moyen.

Question4:

CLOC:

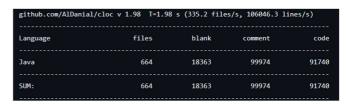
Donnera le nombre de lignes de commentaires. Un seuil sera fixé et si le nombre de commentaire est en dessous de ce seuil, ça nous indiquera que le projet n'est pas suffisamment documenté. Nous utiliserons le programme cloc pour mesurer CLOC.

DC (densité de commentaires) :

Un nombre de commentaires important dans une classe test peut être indicateur de documentation, mais cette métrique ne suffit pas forcément. Nous fixerons un seuil, une DC élevée est un signe que les tests sont potentiellement bien documentés, tandis qu'une DC faible peut indiquer que les tests nécessitent une amélioration de la documentation. Cette métrique sera calculée à l'aide du programme cloc.

Tâche 2 : Mesurer jfreechart

github.com/AlDanial/	cloc v 1.98 T=2.84	s (360.7 file	s/s, 99798.8 lin	ies/s)
Language	files	blank	comment	code
Java	1023	26594	123888	132539
SUM:	1023	26594	123888	132539



Utilisation de cloc dans jfreechart/src

Utilisation de cloc dans jfreechart/src/main

Alexis Boucher 20217120 Marguerite Mireille Camara 20143122

Lien GitHub:

https://github.com/AlexisBoucher/IFT3913

github.com/AlDanial/cloc	v 1.98 T=0.89	s (403.3 file	s/s, 81951.5 lin	es/s)
Language	files	blank	comment	code
Java	359	8231	23914	40799
SUM:	359	8231	23914	40799

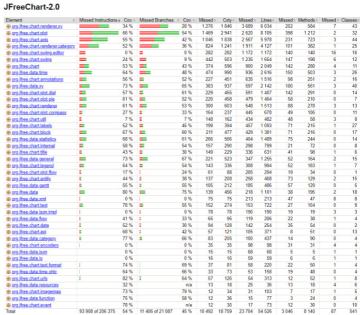
Utilisation de cloc dans jfreechart/src/test

Age en moyenne des fichiers du programme : 289.0 jours Age en moyenne des fichiers tests: 289.0 jours L'age moyen des test sont plus jeunes de 0.0jour

Utilisation de age.java dans jfreechart

Total, 39771, 27, 5.294363, 516

Dernière ligne de l'utilisation de tls.java dans jfreechart



Utilisation de jacoco dans jfreechart

Métrique 1 : Couverture de code

- Utilisation de JaCoCo pour collecter des données de couverture de code.
- Exécution des tests unitaires JFreeChart avec JaCoCo activé.
- Extraction des données de couverture pour chaque classe et package

Métrique 2 : Nombre de lignes par classe

- Utilisation de l'outil CLOC (Count Lines of Code) pour collecter les données sur le nombre de ligne par classe.
- Analyse des résultats de CLOC pour obtenir le nombre de méthodes par classe.

Métrique 3 : Age moyen des fichiers

- La métrique âge recueille les données sur l'âge moyen des fichiers.
- Les données recueillis ne nous permettent pas de conclure concernant l'état de mise à jour des classes car la date du projet reste celle de l'ajout de JFreeChart à notre projet ou au Git

Métrique 4 : Tls

- Cette métrique recueille les données :
 - o Nombre de lignes qui ne sont pas des commentaires dans une classe
 - Nombre d'assertion dans une classe
 - o Tcmp: qui suggère des classes complexes selon un seuil
 - Nombre de test par packages
- Les moyennes sont calculées
- Une analyse des résultats est faite en fonction de la complexité du projet JFreeChart

Pour chaque métrique nous exécuterons des programmes appropriés et collecteront les données pertinentes. Les données collectées seront stockées dans des fichiers CSV ou d'autres formats lisibles comme le .txt.

Alexis Boucher 20217120 Marguerite Mireille Camara 20143122 Lien GitHub: https://github.com/AlexisBoucher/IFT3913

GitHub est utilisé pour fournir l'accès à nos programmes et données collecté, lien du répertoire en tête de page.

Tâche 3 : Réponses aux questions

Question 1: Pour PCC, nous avons fixé un seuil de 60% pour pouvoir conclure qu'il y a assez de test, après analyse avec l'outil Jacoco nous concluons qu'il pourrait y avoir plus de tests car le pourcentage de couverture est de 54% ce qui es inférieur à 60%. Pour TPP nous avons un seuil a 600 et nous observons une moyenne de 516. Pour TPC nous avons un seuil à 40 et nous observons une moyenne de 27, ce qui nous indique qu'il n'y a pas assé de tests

Question 2 : PCC montre qu'il pourrait y a avoir plus de test, (que des ligne de code on été ajouté après que les tests ait été créé) mais nous ne pouvons pas conclure que les tests sont à jours par rapport au reste du code du projet JFreeChart car les données que nous avons collectées ne sont pas représentatives. Cela est dû au fait que le GitHub n'a pas gardé les dates exactes. Dans les normes nous auront fixé un seuil de 30 jours d'âge moyen, tous les fichiers tests vieux de 30 jours et plus que la classe correspondante, auraient été classé non à jour. Si plus de 50% des fichiers étaient classées vieux cela indiquerait que les tests ne sont plus à jour. Pour la mesure de NCH nous avons manquer de temps et ne l'avons pas évalué. Puisque nous n'avons qu'une mesure valide nous ne pouvons pas conclure, mais en raison du résultat donner par jacoco nous pensons que les tests ne sont pas à jour.

Question 3: Nous avons mis un seuil à 40% du code pour la métrique DC, nous observons une densité de code de test de 40 799/132 539 ce qui correspond a un peu plus de 30%. Pour la métrique tcmp nous avons mis un seuil de 10 tloc/tassert. Nous voyons que tcmp moyen est d'environ 5,3. Sur cette base, les tests ne sont pas trop complexes.

Question 4: Sur 81951 lignes recueillis par la métrique Cloc au total dans le projet, nous avons 39771 qui ne sont pas des commentaires donc plus de 50% des lignes sont des commentaires cela peut potentiellement indiquer une densité de commentaire élevé dans le fichier. D'autres analyses sont nécessaires pour donner une conclusion plus précise mais sur la base de cette analyse nous pouvons affirmer que le code est potentiellement assez documenté.