Auteurs:

Kanty-Louange Gakima, matricule: 20184109

Yann-Sibril Saah, matricule : 20061840

IFT3913 – Automne 2023

TP2

**RAPPORT**

**Objectif :** 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.

**Questions :**

**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?

**Choix des métriques à utiliser pour l’analyse :**

|  |  |  |
| --- | --- | --- |
| M1 | LOC | Nombre de lignes de codes y compris les commentaires |
| M2 | DC | Densité de commentaires (pourcentage de lignes qui sont des commentaires) |
| M3 | DET | Durée d’exécution de tous les tests |
| M4 | CLOC | Nombre de ligne de commentaires |
| M5 | NB (Nombre de bugs) | Nombre de bugs |
| M6 | TCC (Taux de couverture de code) | Pourcentage du code qui est couvert par des tests |
| M7 | CC (Complexité́ cyclomatiquede McCabe) | Mesure le nombre de chemin linéairement indépendants |
| M8 | RCT (Ratio taille code/taille test) | Ratio entre le nombre de lignes de tests et le nombre total de lignes de code. |
| M9 | TPR (Taux de de problèmes résolus) | Indique le pourcentage de problèmes résolus parmi ceux détectés (rate of issues) |
| M10 | NEC | Nombre d’erreurs dans le code |
| M11 | PDT (pourcentage de duplications dans les tests) | Indique le taux de tests dupliqués dans les tests |

**Réponse aux questions et analyse :**

[Lien des captures d'écran de l’analyse sur sonarQube](https://viridian-scourge-342.notion.site/TP-IFT3913-2-cb65a46cd3634b8a9b3624a3eb3ad200)

|  |  |  |
| --- | --- | --- |
|  | **Métriques utilisées** | **Interprétation et analyse des résultats** |
| Q1 | **TCC(54%),**  **RCT (40859/92066\*100=44,38%)** | Un taux de couverture de code de 54% indique que seulement la moitié du code source est couverte par des tests. Cela signifie qu'environ 46% du code n'est pas testé.  D'autre part, le ratio de taille du code par rapport à la taille des tests, qui est de 44.38%, indique que les tests ne couvrent qu'environ 44.38% de la taille totale du code source.  En combinant ces deux métriques, il est clair qu'il y a une bonne partie du code qui n’est pas testée, donc en conclusion il n’y a pas assez de tests. |
| Q2 | **TCC (54%), TPR(633/633\*100=100%), NEC(0%)** | 54% du code source est couvert par des tests. Cela signifie que seulement un peu plus de la moitié du code est testé, mais on a quand même 46% du code qui n’est pas testé, Un TPR de 100% et NEC (0%) indique que tous les problèmes identifiés ont été résolus.  Bien qu’il n’y ait pas d'erreurs dans le code et le fait que tous les problèmes identifiés aient été résolus, une grande partie du code n'est pas testée, ce qui indique que les tests ne sont pas à jour avec le reste du code. |
| Q3 | **CC (2689), DET(2s), PDT (9.6%)** | Plus la complexité cyclomatique est élevée, plus le code est complexe et difficile à comprendre et à maintenir. Une valeur de 2689 indique une complexité cyclomatique très élevée, suggérant que les tests sont structurés de manière très complexe. Les tests s’exécutent en 2 secondes ce qui peut indiquer que les tests ne sont pas trop complexes bien que ce n’est pas suffisant. Quant au pourcentage de duplications de 9.6% il est assez faible, cela est positif car ça indique que le code des tests est relativement unique et ne contient pas beaucoup de répétitions  En résumé, même si les tests s’exécutent rapidement et qu’il n y a pas beaucoup de répétions, le complexité élevé indique que les tests sont assez complexes à comprendre. |
| Q4 | **CLOC(5584),DC(12%)** | Un nombre élevé de lignes de commentaires (5584 lignes) indique que les tests sont accompagnés d'une quantité significative de documentation explicative. Et DC = 12% signifie que le 12 des lignes de code dans les tests sont des commentaires, ce qui indique un niveau moyen de documentation.  La densité de commentaires de 12% indique qu'il y a des zones dans lesquelles la documentation pourrait être améliorée, donc en conclusion les tests sont bien documentés mais il y a des améliorations à faire. |

**Conclusion de l’analyse :**

En combinat toutes ces métriques on remarque que l’analyse de jfreeChart nécessite des améliorations, il est essentiel d'augmenter la couverture de code en ajoutant des tests qui couvrent le reste du code, simplifier la complexité des tests en refactorisant les tests, en les découpant en unités plus petites pour réduir les dépendances complexes entre les différentes parties des tests.

Et pour améliorer la documentation des tests, il faudrait ajouter des commentaires de manière ciblée et précise, là où ils sont nécessaires, sans en abuser.

**NB :** Le travail a été remis en retard de deux jours, à l’autorisation du professeur à cause des problèmes rencontrés lors de la configuration d’outils d’analyse, voici une capture d’écran des courriels échangés. Merci pour votre compréhension

A screenshot of a email

Description automatically generated