IFT3913 QUALITÉ DE LOGICIEL ET MÉTRIQUES – AUTOMNE 2022 – TRAVAIL PRATIQUE 2

MISE EN CONTEXTE

Vous allez concevoir (ou plutôt: compléter la conception) et exécuter un plan d'évaluation de qualité du projet JFreeChart, en utilisant le cadre Objectif-Question-Métrique (*Goal Question Metric – GQM).* JFreeChart est une bibliothèque libre de graphiques pour Java : https://www.jfree.org/jfreechart/ créée par Dave Gilbert il y a 22 ans. Son code est disponible à https://github.com/jfree/jfreechart.

G : Votre objectif est d'analyser la dernière version[1] du code de la branche *master* du JFreeChart pour évaluer son niveau de maintenabilité du point de vue du chef du projet.

Vous avez décomposé l'objectif à 4 questions, selon les sous-caractéristiques de maintenabilité définis par la norme ISO 25010 : facilité d'analyse (Q1), facilité de modification (Q2), stabilité (Q3) et facilité de test (Q4).

Q1 : Le niveau de documentation des classes est-il approprié par rapport à leur complexité ?

Q2 : La conception est-elle bien modulaire ?

Q3 : Le code est-il mature ?

Q4 : Le code peut-il être testé bien automatiquement ?

NB : Lisez tous les tâches ci-dessous avant de commencer.

TÂCHE 1 (40%)

Complétez ce plan GQM en opérationnalisant les questions avec des métriques.

Vous pouvez utiliser un maximum de 9 métriques uniques globalement et un minimum de 2 métriques par question (une métrique peut contribuer à plusieurs questions).

Les questions sont un peu de haut niveau et vagues. C'est exprès! C'est à vous à trouver une combinaison des métriques pour chaque question. Vous pouvez utiliser les métriques du cours et vous pouvez en proposer des autres. Consultez l'annexe pour inspiration.

Vous devez expliquer votre raisonnement et motivation pour le choix de chaque métrique. Pourquoi vous pensez que chacune est appropriée (*condition de représentation*) et rentable[2]? Comment prévoyez-vous la mesurer?

NB : Lisez aussi le tâche 2. Les métriques que vous proposez doivent implémentables dans les délais du projet.

NB : Lisez aussi le tâche 3. Quand peut-on dire que la réponse à une question est « oui » ?

[1] Juin 18, 2022 : https://github.com/jfree/jfreechart/tree/b62b85d0d51c9eede3295d09db030a6105978c39
[2] La facilité de mesure peut très bien être un critère, mais le critère principal est la condition de représentation.

TÂCHE 2 (30%)

Mesurez JFreeChart selon votre plan. Votre procédure de mesure doit être automatisée. Vous pouvez utiliser n'importe quel langage de script et outil externe (sauf le code des autres étudiants), y compris les programmes du TP1, les outils que nous avons discuté en classe (cloc, ckjm, …) et autres. Cependant, nous n'offrons pas de conseils sur l'utilisation des outils externes.

Vous devez créer votre propre implémentation pour au minimum une métrique (même si elle est simple).

Décrivez brièvement dans votre rapport votre procédure de mesure.

Expliquez dans votre readme.txt comment nous pouvons accéder à vos scripts et programmes, ainsi qu'aux données que vous avez collectées.

TÂCHE 3 (30%)

Répondez aux questions Q1, Q2, Q3, Q4 en vous basant sur vos données collectées.

Expliquez comment vous avez combiné les différentes métriques pour répondre à chaque question. Le cas échéant, vous pouvez considérer que la réponse pour une question est "oui" globalement si elle l'est pour 90% de tous les cas particuliers (par exemple pour regrouper les données pour tous les classes pour Q1).

En vous basant sur vos réponses aux questions, proposez une évaluation du niveau de maintenabilité du JFreeChart. Est-ce qu'une image claire émerge? Sinon, pourquoi?

PRÉCISIONS GLOBALES

La liste se poursuit à la page suivante, lisez-la en entier !

1. Travail à remettre le **vendredi 21 octobre 23h59** via StudiUM. Aucun retard ne sera accepté.
2. Le livrable le plus important est le **rapport**, en format PDF. Assurez-vous de communiquer clairement en français, la qualité de votre rapport est très importante. Un mauvais rapport pourrait causer une déduction très significative.
3. Travaillez en groupes de 2. **Aucune soumission individuelle ne sera acceptée.** Soumettez un ZIP nommé comme suit : *prenom1_nom1_prenom2_nom2.zip*
4. Le membre de l'équipe qui ne soumet pas le fichier Zip principal de la soumission doit soumettre un zip (même nom) avec le fichier readme.txt, qui devrait contenir les noms de l'équipe.
5. Les rapports doivent compter au maximum 2 pages, y compris toutes les figures et références. Rapports de plus de 2 pages vont être éliminés d'office (par défaut).
6. Votre fichier ZIP doit aussi contenir un readme.txt avec liens vers tous ressources nécessaires (y compris votre repositoire git).

7. Comme en TP1, vous devez utiliser un **repositoire git** pour stocker votre code (idéalement, le même repositoire que TP1). Vous pouvez utiliser n'importe quel service gratuit comme Github, Bitbucket, et autres (quelques-uns vous permettent de créer des comptes académiques avec votre courriel @umontreal.ca). Utilisez le repositoire pour collaborer avec votre coéquipier.
8. Vous devez mettre dans votre repositoire git tout script, ou fichier de configuration que vous avez utilisé. En fait, votre repositoire doit contenir assez des détails pour nous convaincre que vous avez vraiment effectué le travail vous-même.
9. Nous allons examiner l'historique de votre repositoire pour nous assurer que tous les deux coéquipiers ont travaillé sur le TP et que votre code n'est pas plagié. Un historique de commit plausible devrait contenir de nombreux petits commit, chacun avec un message de commit approprié. **Faire juste quelques commit massives proche à la date limite pourrait entraîner une déduction considérable.**

ANNEXE

Quelques métriques potentiellement utiles :

- LOC, CLOC, NCLOC
- Ratio taille code / taille test
- DIT, NOC, WMC, CBO, LCOM1/2, RFC
- DC (densité de commentaires)
- NVLOC, CSEC (voir TP1)
- NOM (nombre de méthodes)
- CAC (nombre d'associations auxquelles participe une classe)
- NEC (nombre d'erreurs)
- NAT (nombre d'attributs)
- AGE (age d'un fichier)
- NCH (nombre de commits dans l'historique d'une classe)
- TPC (tests par classe)
- PMNT (pourcentage de méthodes non testés)
- NBAC (nombre de bugs attribuable à une classe)
- CC (complexité cyclomatique d'une méthode)

et autres…