IFT3913 QUALITÉ DE LOGICIEL ET MÉTRIQUES – AUTOMNE 2023 – 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 23 ans. Son code est disponible à https://github.com/jfree/jfreechart.

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

Supposez que vous avez décomposé l'objectif à 4 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?

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

TÂCHE 1 (40%)

Complétez ce plan GQM en opérationnalisant les questions avec des métriques. Vous avez un «budget» de 15 métriques maximum et vous êtes encouragés à réutiliser des métriques pour plusieurs questions. Il n'est pas raisonnable de se baser sur une seule métrique pour répondre à une question. Essayez donc de baser votre analyse sur au minimum de 2-3 métriques par question.

Les questions sont un peu de haut niveau et vagues**. C'est exprès!** C'est à vous à trouver une combinaison de métriques qui pourraient vous aider à répondre aux questions. Vous pouvez utiliser les métriques du cours et vous pouvez en proposer d'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 la tâche 2. Les métriques que vous proposez doivent être implémentables dans les délais du projet.

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

[1] Version 1.5.4 : https://github.com/jfree/jfreechart/tree/v1.5.4

[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).

Incluez les valeurs des différentes métriques proposées dans votre rapport et décrivez brièvement 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. Dans le cas où vous devez combiner des cas particuliers pour donner une réponse globale, vous pouvez définir une stratégie d'agrégation (par exemple : «si la réponse est oui pour 90% des cas, la réponse sera considérée oui globalement» ou «la réponse est considérée oui si la moyenne est supérieure de 80%»). Expliquez vos choix!

En vous basant sur vos réponses aux questions, proposez une évaluation de la facilité d'analyse 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 27 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 travail doit être soumis par un seul membre de l'équipe. Celui 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** 4 pages, y compris toutes les figures et références. Rapports de plus de 4 pages vont être éliminés d'office (par défaut). (La taille de 4 pages est un budget, pas un objectif à atteindre.)
6. Votre fichier ZIP doit aussi contenir un readme.txt avec liens vers tous ressources nécessaires (y compris votre répertoire git).

7. Comme en TP1, vous devez utiliser un **répertoire git** pour stocker votre code (idéalement, le même répertoire 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 répertoire pour collaborer avec votre coéquipier.
8. Le répertoire git doit être **privé**.
9. Ajoutez le compte « **OussamaSghaier** » comme collaborateur de votre répertoire.
10. Vous devez mettre dans votre répertoire git tout script, ou fichier de configuration que vous avez utilisé. En fait, votre répertoire doit contenir assez de détails pour nous convaincre que vous avez vraiment effectué le travail vous-même.
11. Nous allons examiner l'historique de votre répertoire 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)
- TLOC, TASSERT, TCMP (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)
- TPP (tests par package)
- PMNT (pourcentage de méthodes non testés)
- NBAC (nombre de bugs attribuable à une classe)
- CC (complexité cyclomatique d'une méthode)

et autres…