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

Pour évaluer la première question on commence par compter le nombre de méthodes de tests qu’il y a par rapport au nombre de méthodes fonctionnels il y a dans le projet. Ainsi on établie la première métrique qui est le rapport de tests par rapport au nombre de méthodes fonctionnelles. De cette manière nous arrivons à avoir un rapport 1 pour 1 qui démontrerait grossièrement que chaque méthode fonctionnel du projet a un test. Parfois il est nécessaire de faire attention avec cette métrique car elle n’élicite pas une couverture de code mais en fait une estimation.

Nous pouvons répondre à la deuxième question à l’aide de la métrique *méthodes de test / méthodes fonctionnelles*. Présumant que chaque méthode fonctionnel aie besoin que d’une méthode de test on peut estimer que les tests sont à jour avec le restant du code s’il y a au moins un test par méthode de l’application.

En utilisant simplement

Le nombre d’assertions par méthode de test serait une bonne métrique de complexité.

Quote du projet que l’on se sert.

LOC (Lines of code): It counts the lines of count, ignoring empty lines and comments (i.e., it's Source Lines of Code, or SLOC). The number of lines here might be a bit different from

Les métriques et l’explication d’implémentation :

1. LOC, CLOC, NCLOC  
   Utilisation d’une librairie pour mesurer NCLOC et mesure CLOC par implémentation propre. CLOC mesurer à l’aide d’un filtrage regex pour enlever les lignes de commentaires et ceci soustrait au nombre de lignes dans la classe CLOC = LOC - NCLOC.
2. TPC (Tests par classe)  
   Utilisation d’une librairie pour le nombre de méthodes. Filtrage de ces méthodes à l’aide de regex pour déterminer les méthodes qui ont au moins 1 assertions dans la méthode.
3. Ratio taille code / taille tests  
   Mesure de la taille du code à l’aide d’une librairie, mesure de la taille des tests à l’aide d’une mesure de la taille de code des méthodes de tests (celles qui contiennent des assertions).
4. PMNT (Pourcentage méthodes non-testées)  
   À implémenter !!
5. Nombre de commits dans l’historique d’une classe  
   Utilisation de la ligne de commande dans le projet pour utiliser les outils de ligne de commande de git présumément installés sur la machine. Ainsi on somme les commits associés au fichier.
6. Âge d’un fichier (dernier update)  
   Utilisation de la ligne de commande dans le projet pour utiliser les outils de ligne de commande de git présumément installés sur la machine. On mesure la dernière date d’un commit sur un fichier avec un log de la date du dernier commit.
7. RFC  
   Mesurer à l’aide de la librairie.
8. WMC  
   Mesurer à l’aide de la librairie.
9. Densité de commentaires   
   Mesure CLOC / LOC.

the original file, as we use JDT's internal representation of the source code to calculate it.