# Une image contenant texte Description générée automatiquementRapport projet 4 – IFT 3913

Auriane EGAL 20123018

Anass Eddaoudi 20173147

## Tâche 1 :

Le programme proto a été construit sous forme à produire les données puis les mettre dans le fichier CSV en code Java. Premièrement il demande à l’utilisateur le nom du *path*, soit le lien URL pour le git qu’on veut analyser. Puis après l’utilisateur devra rentrer le nom du *dossier* en question (normalement la dernière chaine de caractère dans le *path*).

Le programme va faire appel en premier à la méthode *version* pour prendre les id version. Pour cela on fait appel à l’invite de commande (sous Windows) sur lequel on fait appel à *git rev-list –all*. Une fois exécuté, on récupère la ligne que l’on met dans une *ArrayList*. Si le terminal affiche un code d’erreur différent de 0 alors ça signifie qu’on a déjà le dossier en question ou autre.

Après avoir fini, le programme va faire de même pour la récupération du nombre de classes en fonction de l’id. On fait appel à une autre fonction *nmbClasses*. La fonction va utiliser le même procédé que la première fonction. On va juste vérifier chaque sorties (lignes) si c’est une classe java, et si c’est le cas on incrémente notre compteur.

Une fois les données récupérer, on créer notre CSV. La première ligne contient les id séparer par des virgules et la deuxième ligne contient le nombre de classes associées qui est aussi séparer par des virgules.

*Note : notre programme donne en sortie 2 lignes lorsque qu’on l’ouvre dans l’IDE (Visual Studio Code en l’occurrence). Mais lorsque qu’on l’ouvre dans Excel on se retrouve avec 7 lignes. Donc on suppose que Excel produit cette erreur.*

## Tâche 2 :

Toutes les modifications ont été faites dans la fonction *nmbClasses*, soit entre les commentaires :

// Pour la tache 2

… Code rajouté

// Fin pour la tâche 2

Principalement on a rajouté deux *ArrayList* qui permettent de sauvegarder les données. Elle fait appel à notre ancienne implémentation du TP1 (qu’on a importé).

En sorti on obtient deux nouvelles colonnes étant la moyenne de la métrique WMC et la moyenne de la métrique de la classe BC.

## Tâche 3 :

Nous avons appliqué *proto* a l’entièreté du projet JFREECHART et nous avons inclus le fichier CSV dans le zip. Nous avons même un fichier Excel ou ces mêmes données sont normalisé.

## Tâche 4 :

Graphique 1 : nombre de classe en fonction des versions

On peut voir sur le graphique que le projet tournait autour de 1000 fichier java (classe) puis plus on avance dans les versions plus on a moins de classes dans le projet. On peut aussi observer un pic descendant près de l’origine ce qui pourrait porter à croire que le projet a eu droit à un *restart* a un moment donner.

Graphique 2 :

On voit sur le graphique que la valeur MWMC diminue plus les versions avance jusqu’à atteindre 0. On retrouve un pic de complexité a peu près aux même version ou on retrouve le pic descendant du graphique 1.

Graphique 3 :

Dans ce dernier graphique qui met en relation MCBC en fonction de la version de JFREECHART on peut voir que plus on avance dans les versions moins la valeur de MCBC a de poids jusqu’à atteindre 0.

## Tâche 5 :

**Hypothèse : Il existe une corrélation entre les métriques NC et MWMC**

Pour évaluer cette hypothèse on peut superposer les graphiques NC en fonction des versions et MWMC en fonction des versions.

Si on analyse ce graphique ainsi que les courbe de régression linéaire respective on peut voir que les 2 sont décroissante mais n’ont pas le même ordre de grandeur. Si on compare les fonctions :

NC en fonction des versions : y= -0,019x+1012

MWMC en fonction des versions : y= -0,0039x+12,74

On peut alors voir que les 2 sont des fonctions négative mais l’ordre de grandeur est complètement disproportionné.

Si on souhaite explique les chiffres à l’aide d’un contexte réelle on peut dire que plus le projet a de version (donc est améliorer et/ou simplifier et/ou optimiser) le nombre de classe et de fichier impliqué diminuent ainsi en est pour la moyenne de la métrique WMC. Dans ce scénario on peut donc dire qu’il y a de moins en moins de complexité plus le projet avance.

On peut alors conclure qu’il existe certes une corrélation mais qu’elle n’est pas linéaire.

**Hypothèse : Il existe une corrélation entre les métriques NC et MCBC**

Pour évaluer cette hypothèse on superpose les graphiques en fonction des versions.

Ici nous pouvons voir que les 2 graphique ont une courbe de régression négative mais encore une fois dans un différend ordre de grandeur.

NC en fonction des versions : y= -0,019x+1012

MCBC en fonctions des versions : y=-0,0004x+0,325

Si on souhaite expliquer les chiffres à l’aide de leur contexte on pourrait considérer le fait que plus de version de JFREECHART ont été publié plus les classes se simplifiait (hypothèse précédente) et moins il était important de commenter de petites classes. Ou au vu de la valeur de MCBC il est possible que les développeurs aient un nouveau moyen de donner l’information à propos de leur code.

On peut alors conclure qu’il existe une corrélation puisque les trois graphiques ont le même point tournant autour de 1200 commit mais la corrélation n’est pas linéaire.