Master 2 Informatique
Spécialité IAGL

Rapport de projet OPL

Thème : Pull Request Engineering

Vérification automatique des règles de codage lors d'une Pull Request

Antoine PETIT
antoine1.petit@etudiant.univ-lille1.fr

Rémi KRUCZEK
remi.kruczek@etudiant.univ-lille1.fr

11 Octobre 2016

Plan

1 Introduction

Le contexte

Le développement d'un projet informatique est souvent l'oeuvre d'un équipe de plusieurs développeurs qui ajoute chacun leur tour, leur contribution. Avec l'augmentation de projet open source et d'outils permettant leur diffusion, il a émergé un tendance qui consiste à proposer ses contributions aux projets d'autres personnes.
Ces contributions peuvent se présenter sous la forme d'une pull-request. Ce sont des modifications de code qui sont proposées pour le projet. Elles peuvent être discutées grâce à des commentaires, pour au final être acceptées ou refusées.

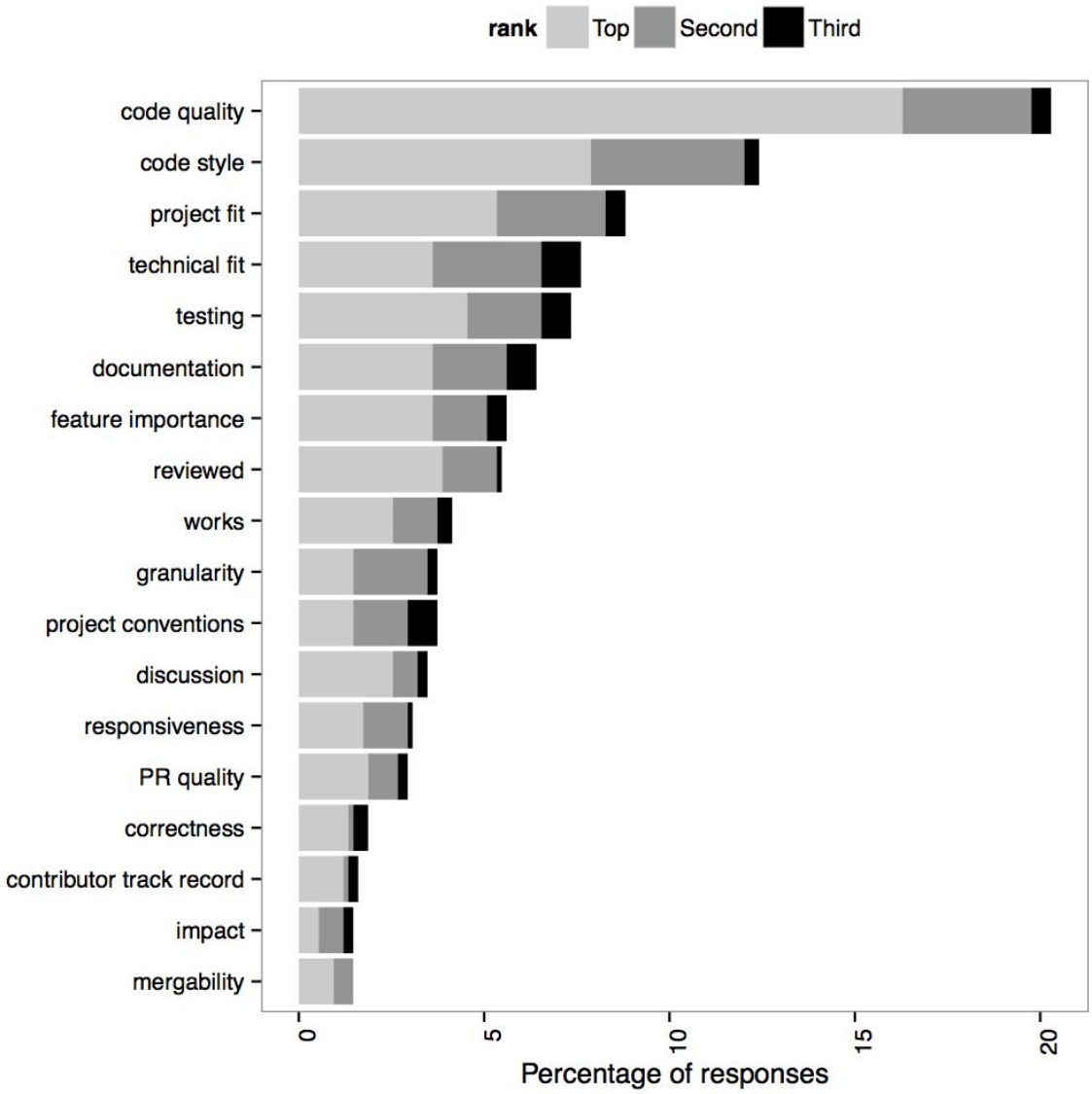

Figure 1 : Étude qui montre les raisons qui influencent l'intégrateur à accepter une pull-request.

Une étape important dans le choix final d'une pull-request est donc l'étude de celle-ci. L'acceptation d'une pull-request peut être influencée par plusieurs critères. Le critère le plus important peut être l'intérêt de la fonctionnalité proposé, on peut aussi vérifier la qualité du code, la qualité des commentaires, la présence de tests, la présence d'une documentation…

Toutes ces vérifications doivent donc être effectuées par un responsable du projet, qui sera au courant des caractéristiques du projet et qui va vérifier que ces caractéristiques sont respectées dans la pull-request. Ces étapes peuvent prendre du temps, et cela peut se transformer en perte de temps si la pull-request est finalement refusée.

Le but

Notre objectif était de mettre en place un outil qui doit servir à l'intégrateur et lui permettre de gagner du temps sur l'étude des pulls-requests soumises à son projet.
Pour cela l'outil va se consacrer à la vérification des règles de codage sur le code soumis. Car comme le montre la figure 1, les règles de codage sont très prise en compte par les intégrateurs. Lors d'une pull-request, l'outils va automatiquement vérifier que les règles de codage définis dans le projet sont respectées.

L'exécution de programme de vérification de règle de codage peut prendre énormément de temps sur de gros projet. Et lancer ce programme a chaque pull request peut être problématique sur des projets populaires.
La caractéristique de l'outils proposé est donc de n'effectuer ses vérifications que sur le nouveau code ajouté par la pull request.
L'évaluation de l'outil sera donc fait sur la comparaison de son temps d'exécution sur des pull request, et sur le code complet d'un gros projet.

Dans un premier temps nous vous présenterons l'environnement et le travail technique réalisé pour mettre au point cet outil. Enfin nous vous présenterons la mise en place de l'évaluation de l'outil et de ses limitations.

2 Travail technique

Le but du projet est donc d'automatiser l'exécution d’un programme de vérification de règles de codage sur le code proposé dans une pull-request.

Les outils

- **Java** : Le projet est réalisé avec le langage Java, qui est un langage orienté objet.
- **Checkstyle** : Programme de vérification de règles de codage. On peut définir les règles à vérifier grâce à un fichier de configuration de type XML.
- **Github** : Application web qui permet de stocker le code d’un projet. Elle met en place des fonction de versionning et de contribution au projet grâce aux pull request.
- **WebHook** : C’est une fonction qui permet l’envoie automatique d’un requête avec le déclenchement d’une action. Pour le projet nous nous en servons pour récupérer les information de la pull-request lors de la création de celle-ci.
- **Json** : C’est un format de données. L’API de github utilise uniquement ce format.
- **Ngrok** : Programme qui permet de mettre en place un tunnel entre un ordinateur et une adresse web. A terme ce programme ne devrait plus être utilisé.

L’utilisation

Une adresse web est créé grâce à Ngrok et doit être connecté au port 8000.
L’adresse web générée doit être ajoutée sur la webhook du compte github.
Après avoir lancé l’outil il indiquera de renseigner le token (de l’intégrateur) donnant les droits aux projets Github qui devront être surveillé. Ensuite l’outil attend un webhook de la part Github.

Lors de la création d’une pull-request de la part d’un contributeur, Github envoie donc une requête sur l’adresse web, qui est, au final, récupérer par l’outil.
L’outil récupère alors l’ensemble des fichiers composant le “diff” de la pull-request puis récupère le fichier Checkstyle du projet.
L’outil lance ensuite le programme Checkstyle sur tous les fichiers.
Le résultat du Checkstyle est enfin ajouté à la pull request sous la forme d’un commentaire.
Ainsi l’intégrateur sera en un coup d’oeil si il peut vérifier les autres points de la pull-request.

3 Evaluation

Avantages

Sur de gros projets, qui contiennent de nombreux fichiers de code, l'exécution de vérification peut être redouter car l'entièreté du code sera vérifiée, par conséquent, pour les projets qui n'utilisaient pas jusque là les vérifications, la présence de nombreuses erreurs de style peut devenir visible.
Cet outil permet de se concentrer uniquement sur le code ajouté, ainsi les intégrateurs et développeurs savent que le code qu'ils modifient est correct par rapport aux règles du projet. Néanmoins cet outil n'est pas suffisant si on veut faire une refonte totale du code afin qu'il corresponde aux nouvelles normes. Dans ce cas il faut d'abord faire une vérification entière du code via un checkstyle afin de repérer les zones posant problèmes.
De plus, le fait d'exécuter la vérification seulement sur les fichiers du "diff" de la pull-request le temps d'exécution des vérifications sera plus court qu'en vérifiant l'ensemble du code.

Pour montrer son intérêt, l'idée est de comparer le temps d'exécution de la solution sur un projet conséquent. On verra alors la différence entre le temps d'exécution d'une vérification sur tout le code d'un projet, et seulement sur les dernières modifications du code. Malheureusement dans le cadre de ce rendu nous avons été incapable de mettre en place ce jeu de test.

4 Limitations

L’objectif principale de la solution est de limiter les vérification au maximum, pour gagner du temps mais également encourager les développeur à revoir leur code. Malheureusement, la solution ne permet pas de prendre en compte seulement les nouvelles lignes de code de la pull request.
En effet Checkstyle suit la logique d’un fichier java, et prendre des lignes de code qui n’ont aucune relation entre elles lève des exceptions.

La solution se base donc sur les fichiers qui contiennent le code modifier. Lors d’une modification de code, c’est tout le fichier qui est analysé. Ce partie pris permet de prendre en compte toutes les règles possibles au niveau de checkstyle. Notamment pour les règles d’indentation, où le fichier entier est nécessaire.

.

Améliorations

Une fois complet le programme devrait être un serveur distant auquel s’inscrivent les projets. Idéalement le programme devrait être capable de tourner uniquement sur le “diff” afin de ne pas avoir de retour supplémentaire que ceux liés aux changements, malheureusement on ne peut pas appliquer ça avec Checkstyle.

Une autre amélioration serait de faire que les commentaires invalidant la pull-request précisent les règles qui n’ont pas été respectées. Ils pourraient également indiquer la ligne sur laquelle le problème à eu lieu, grâce à l’outil intégré dans l’API Github.

5 Conclusion

Nous avons créé la première brique d’un outil qui permet de vérifier les règles de codage sur chaque nouvelle ligne de code ajoutée au projet. Ce genre d’outil pourrait avoir un intérêt pour les gros projets qui souhaitent faire en sorte que le nouveau code respect une certaine norme. Les erreurs sur l’ensemble d’un projet pouvant cacher celle d’une évolution.

Cet outil serait d’autant plus intéressant s'il était couplé à un outil de correction automatique de syntaxe afin de corriger à la réception du code les erreurs de ce dernier. Ainsi le temps ne serait plus uniquement gagner du côté de l’intégrateur mais aussi au niveau de développeur.