Thomas Neyraut
1812273

LOG6302 : Ré-ingénierie du logiciel

Rapport d'avancement N°4

INTRODUCTION

Ce document présente l'avancement qui a été effectué au cours de la septième et huitième semaine de projet. Durant ces deux semaines, l'objectif était de générer à l'aide de javacc et GraphViz le diagramme des dominateurs, le diagramme des post-dominateurs et le diagramme des définitions valides (in, gen, kill et out) d'Eclipse JDT core. Dans une première partie, nous présentons l'ensemble de la méthodologie. Dans une seconde partie, nous présentons les avancées et les résultats que nous avons réalisés au cours des deux dernières semaines, ainsi que les difficultés que nous avons pu rencontrer.

I. METHODOLOGIE

Pour réaliser ces différentes tâches, nous avons dans un premier temps modifier une toute petite partie du visiteur CFGVisitor permettant d'extraire le diagramme de flux de contrôle. Nous avons aussi dû modifier les classes déjà implémentées suivantes :

- FinallyStructure.java,
- Switch.java,
- Structure.java,
- If.java,
- While.java,
- Case.java,
- Continue.java,
- Variable.java,
- Do.java,
- Catch.java,
- Try.java,
- Throw.java,
- For.java,
- Return.java,
- Break.java,
- Method.java.

Dans chacune de ces classes, nous avons principalement ajouté trois méthodes supplémentaires :

- getDiagramDominateursFormatGraphViz(),
- getDiagramPostDominateursFormatGraphViz(),
- getGraphEntryOutFormatGraphViz().

Ces trois méthodes permettent respectivement de générer partiellement le diagramme de dominateurs, le diagramme des post-dominateurs et le diagramme des définitions valides sous le format GraphViz. Pour générer complètement un diagramme pour une méthode m, le logiciel appelle l'une de ces trois méthodes pour l'instance m. Les fichiers .dot des diagrammes sont générés respectivement via les classes suivantes :

- DominateurGenerator.java,
- PostDominateurGenerator.java,
- GraphEntryOutGenerator.java.

Pour finir, nous avons modifié le fichier JavaParser1_7.java afin d'ajouter des nouvelles options pour exécuter les différentes parties du logiciel. A présent, il est possible d'exécuter notre logiciel avec les lignes de commandes suivantes :

- java Javaparser1_7 path_name_file_java_or_list_file 0 json : permet de récupérer les métriques dans un fichier json,
- java Javaparser1_7 path_name_file_java_or_list_file 0 csv : permet de récupérer les métriques dans un fichier csv,
- java Javaparser1_7 path_name_file_java_or_list_file 1 : permet de générer le diagramme de classes via plusieurs fichiers .dot,
- java Javaparser1_7 path_name_file_java_or_list_file 2 : permet de générer le diagramme de flux de contrôle via plusieurs fichiers .dot,
- java Javaparser1_7 path_name_file_java_or_list_file 3 : permet de générer le diagramme des dominateurs via plusieurs fichiers .dot,
- java Javaparser1_7 path_name_file_java_or_list_file 4 : permet de générer le diagramme des post-dominateurs via plusieurs fichiers .dot,
- java Javaparser1_7 path_name_file_java_or_list_file 5 : permet de générer le diagramme des définitions valides via plusieurs fichiers .dot.

II. RESULTATS ET DISCUSSION

Tout comme pour les diagrammes de classes et les diagrammes de flux de contrôle que nous avons généré lors des semaines précédentes, les trois nouveaux types de diagrammes que nous générons peuvent dans certains cas fournir des fichiers .dot trop lourds pour être lu par GraphViz. De ce fait, nous avons décidés de diviser les diagrammes. Chaque fichier .dot correspond à un fichier .java (donc à une classe java) et il contient au maximum 20 diagrammes de 20 méthodes implémentées dans le fichier .java considéré. Nous avions utilisé cette méthodologie de séparation pour les diagrammes de flux de contrôle lors des semaines précédentes. Malheureusement dans certains cas pathologiques, il est toujours possible d'obtenir des fichiers .dot impossible à ouvrir à cause de leur taille trop importante. Il est donc possible d'obtenir des diagrammes de classes, des diagrammes de flux de contrôle, des diagrammes des dominateurs, des diagrammes des post-dominateurs et des diagrammes des définitions valides impossible à ouvrir. Cela serait dû à la présence de plusieurs méthodes de très grandes tailles dans le fichier .java considéré.

Enfin le langage de programmation java étant très permissif, il est fort probable que n'ayons pas considéré l'ensemble des expressions et instructions possibles. Il est

donc probable de rencontrer dans certains logiciels des expressions et instructions que notre programme ne prend encore en compte.

Conclusion

Pour conclure, nous avons réussi à implémenter un programme nous permettant de générer plusieurs types de diagrammes caractéristiques d'un logiciel. Cependant dans certains cas pathologiques, les fichiers générés peuvent ne pas s'ouvrir. De plus, notre programme ne prend pas encore en compte l'ensemble des expressions et instructions permissent par le langage de programmation java. De ce fait dans les jours à venir, nous allons modifier notre programme pour qu'il puisse considérer le plus d'expressions et d'instructions possibles. Il faudrait aussi chercher à concevoir un système permettant d'éviter que certains fichiers .dot ne puissent pas être ouvrable.