Thomas Neyraut
1812273

LOG6302 : Ré-ingénierie du logiciel

Rapport d'avancement N°3

INTRODUCTION

Ce document présente l'avancement qui a été effectué au cours de la cinquième et sixième semaine de projet. Durant ces deux semaines, l'objectif était de générer à l'aide de javacc et de GraphViz le diagramme de flux de contrôle pour le programme 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

Dans un premier temps, nous avons implémenté un ensemble de classes permettant de représenter les différentes structures pouvant être prises en compte dans un diagramme de flux de contrôle. Les classes implémentées sont les suivantes :

- For.java,
- If.java (nous utilisons cette classe pour représenter les if, les else et les else if),
- While.java,
- Do.java,
- Switch.java,
- Case.java,
- Catch.java,
- Try.java,
- Throw.java,
- Break.java,
- Return.java,
- Continue.java.

Toutes ces classes héritent de la classe Structure. Elles ont donc certaines méthodes et attributs en commun. Nous avons notamment modifié la classe Variable pour potentiellement intégrer dans le diagramme de flux de contrôle la déclaration et l'assignation de variables locales. Nous avons aussi dû modifier légèrement notre classe Method.

Les instances de ces différentes classes disposent toutes d'un attribut id (int) permettant de les distinguer dans le diagramme de flux de contrôle. De plus, toutes ces classes disposent d'une méthode, provenant de leur super classe Structure, qu'elles surchargent et qui permet de construire partiellement le diagramme de flux de contrôle. Pour construire le diagramme dans sa globalité, il faut appeler cette méthode pour chaque instance de la classe Method.

Par la suite, nous avons implémenté un nouveau visiteur CFGVisitor. Ce nouveau visiteur permet de visiter un programme Java et de construire le diagramme de flux de contrôle du programme. Notre programme visite un fichier java à la fois et génère pour chaque fichier java un fichier .dot représentant le diagramme de flux de contrôle correspondant.

Au cours de son exécution, le visiteur va instancier plusieurs instances de la classe Method (qu'il va stocker dans un ArrayList<Method>), puis va instancier plusieurs instances des différentes classes présentées précédemment héritant de la classe Structure. Chacun de ces objets va être associé à une instance de la classe Method via un ArrayList<Structure> ou à une instance d'une des classes suivantes via un ArrayList<Structure> : For, While, If, Do, Switch (que pour des objets Case), Case, Catch, Try. Ainsi à la fin de l'exécution du visiteur, nous disposons en mémoire de notre diagramme de flux de contrôle. Le visiteur dispose d'un attribut id (int) initialement égale à 1 et qui est incrémenté à la valeur supérieure à chaque instanciation d'un objet de type Method ou Structure. Cet attribut id nous permet de différencier avec certitude les différents éléments présents dans un diagramme de flux de contrôle.

Pour finir, nous avons implémenté la classe CFGGenerator permettant de générer le diagramme de flux de contrôle via GraphViz. Cette classe utilise l'arrayList du visiteur contenant l'ensemble des méthodes présentes dans un fichier java donné. Pour chacun des éléments de l'arrayList, la classe CFGGenerator va appeler une méthode renvoyant la représentant du diagramme de flux de contrôle de la méthode considérée sous le format GraphViz. On obtient ainsi un fichier .dot comprenant le diagramme de flux de contrôle de chaque méthode présente dans un fichier java donné.

II. RESULTATS ET DISCUSSION

Nous n'avons pas rencontré de problèmes très complexes et intéressants au cours de ces deux dernières semaines. Nous avons légèrement perdu du temps dans l'implémentation de notre visiteur. En effet, nous avons dû tester plusieurs méthodes afin de déterminer les méthodes les plus adéquates. L'architecture que nous avons mise au point nous a permis d'ajouter facilement de nouvelles structures au cours de notre développement. Nous avons en effet pu ajouter facilement les structures Try, Catch, Throw et Variable. Le comportement spécifique de certaines structures nous a demandé un peu plus de temps d'implémentation. En effet, les break, return et continue sont des cas « particuliers » qui nous ont demandé une attention particulière.

Enfin, l'un des problèmes que nous avons rencontrés est le même que celui que nous avions rencontré pour la génération du diagramme de classes lors de l'étape précédente. En effet à l'étape précédente, certains des fichiers .dot étaient impossible à ouvrir via GraphViz à cause de leur taille. Ici, nous avons le même problème. En effet, si nous nous trouvons en face d'une « God Class » disposant de nombreuses et longues méthodes, le fichier .dot correspondant aux diagrammes de flux de contrôle des différentes méthodes de cette classe est bien trop gros et donc impossible à ouvrir via

GraphViz. Pour résoudre ce problème, nous avons décidé de fragmenter les fichiers .dot sans diviser le moindre diagramme de flux de contrôle. Un fichier .dot ne peut pas contenir plus de 20 diagrammes de flux de contrôle.

Conclusion

Pour conclure, nous avons réussi à implémenter dans les temps un programme fonctionnant parfaitement, et permettant de générer le diagramme de flux de contrôle d'un logiciel développé en java.