ENHANCED COVERAGE REPORTING

Le contexte

Il existe plusieurs outils qui peuvent produire des statistiques sur la manière dont une suite de tests couvre un programme.
La plus part de ces outils ne prennent en compte que la couverture statique du code et se focalisent pas ou peu sur l'aspect dynamique de la couverture

Par exemple ces outils permettent de savoir si une méthode a été appelée dans l'exécution d'un programme mais ne fournissent aucune information par exemple sur le nombre de fois que celle çi a été appelée ou si elle a été appelée par les mêmes paramètres à chaque fois ou des paramètres différents.

L'objectif de ce projet est de développer une technique permettant de d'instrumenter un programe et un suite de test dans le but de collecter des information dynamique en rapport avec la couverture de code

Pour ce travail on s'est proposé de partir d'un cas type simple qui est : « *Pour chaque méthode dans programme donné, combien de fois cette méthode est-elle appelée lorsqu'on lance une suite de test sur ce programme* »

La solution

Pour recueillir cette mesure on a besoin d'instrumenter le code du programme pour placer des sondes dans chaque méthode du programme. Ces sondes permettent de détecter à l'exécution si une méthode est exécutée et incrémenter un compteur à chaque fois que cette dernière est appelée.

La sonde

Pour notre càs une sonde est représentée par un méthode qui est appelée à chaque exécution d'une procédure . Cette sonde est placée statiquement dans chaque classe du programme à l'aide d'un outils d'instrumentation dans ce projet on utilise l'outils spoon.

Structure de stockage

Cette procédure stocke dans une structure de donnée (HashMap pour notre càs) des couples clé valeur , la clé étant le nom d'une procédure de la classe et la valeur un entier qui est incrémenté à chaque fois que cette procédure est appelée.

L’instrumentation

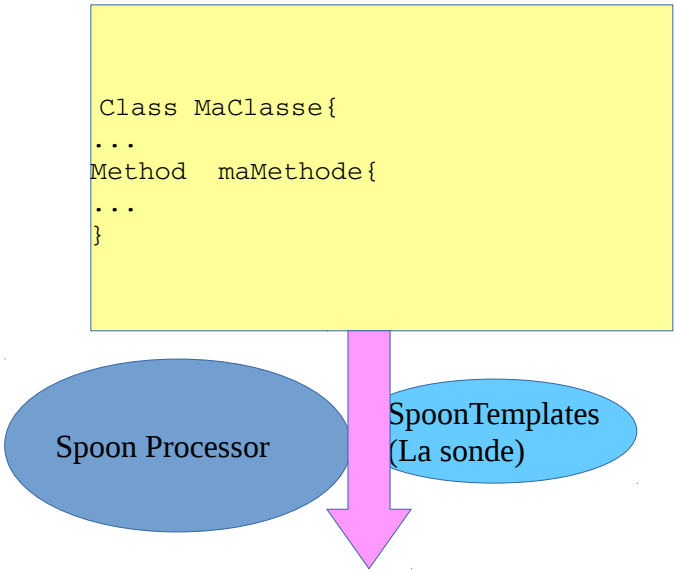
Pour pouvoir ajouter nos sondes, on a utilisé l'instrumentation de code, un outil particulier **Spoon** nous a permis d’instrumenter le code est de rajouter les bouts de code nécessaires.

Spoon est une librairie open-source qui permet l’analyse statique d’un code java , spoon fourni un méta-modèle où chaque élément d’un programme , package class ,méthode peut être accédé et modifié
Spoon prend un code source est retourne un code source transformé. Spoon se base sur une notion centrale de processeur.

Un processeur est une classe java qui s’occupe de l’analyse d’un certain type d'élément dans le méta modèle généré par spoon.

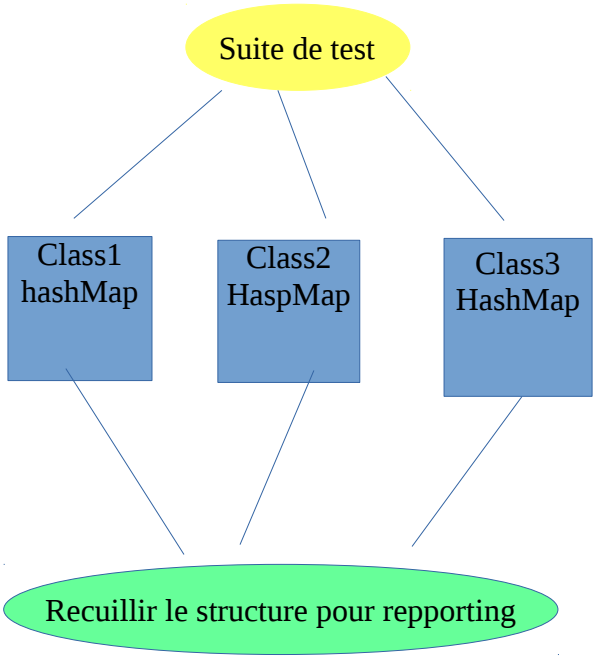
Une autre notion importante de spoon est la notion de template.
Les templates permettent d’ajouter des éléments statiquement analysés ce qui permet d’assurer que ces derniers ne seront pas source d’erreur dans le code où il seront ajoutés.

Pour notre prototype les deux notions processeur template ont été utilisée le schéma suivant permet d’illustrer




```
Class MaClasse{
…
Structure stockage(HashMap)  // ajoutee par spoon

Method maMethode{
..Sonde()//   incrementer la valeur dans la sstructure de stockage
…
Sonde() //procedure ajoutée par instrumentation
}
```



Evaluation

Pour pouvoir tester notre prototype, nous somme partie d'un programme avec une suite de test en Junit, on fournit le code source en entrée à spoon ce dernier génère un code transformé sur lequel on relance les test pour recueillir nos métriques.

Pour cela on crée un clone du projet de départ et on fournit à spoon un chemin de sorti qui est le répertoire source du projet clone ,on obtient un programme sur lequel la partie test reste inchangée avec une partie source du programme transformé pour inclure les sondes.

On lance les tests par Junit et on prévoit une écriture sur la sortie standard pour consulter les structures de donnée qui contiennent les métriques désirées.

Conclusion

La solution proposée est une solution qui utilise l'instrumentation de code donc un ajout statique de code dans du code , cette manière de faire reste très peu flexible et spécialement avec spoon , spoon faisant de l'analyse à statique pour construire son méta-modèle a besoin de tous les dépendances qui se trouvent dans le programme en entrée ce qui peut représenter un coup pour des projets de taille importante.

D'outres techniques tel le tissage dans la programmation aspects pourrait être utilisé en conjonction avec l'instrumentation statique pour beaucoup plus d'efficacité.

Il reste