M2 Informatique AIGLE

**HMIN306**

Évolution et Restructuration

# Rapport de TP

## (TP 4)

**Bachar Rima**,
**Amandine Paillard**

17 janvier 2020

# Table des matières

# Chapitre 1

# TP4 - Réingénierie des logiciels

## 1.1 Exercice 1 - Extraction de la variabilité

Le but de cet exercice est de pouvoir comparer deux versions différentes du développement du même logiciel. Idéalement il aurait fallu le tester sur un projet de plus grande ampleur comme ArgoUML. Cependant après avoir essayé notre programme sur ce logiciel, nous avons constaté que l'exécution prenait un temps certain. Par souci de simplicité nous avons donc exécuté nos analyses sur le code réalisé dans le cadre de ce TP ainsi que sur une version altérée de ce code. Le but de ces altérations étant de rendre rendre les deux projets suffisamment différents pour pouvoir appliquer nos scénarios d'analyse (héritage, nouvelles méthodes...).

Les scénarios ainsi étudiés sont :

— Identifier les classes communes et variables entre deux versions d'un logiciel en se basant sur le nom de la classe [1] ;
— Identifier les classes communes et variables entre deux versions d'un logiciel en se basant sur la liste des méthodes de la classe ;
— Identifier les classes communes et variables entre deux versions d'un logiciel en se basant sur la déclaration d'héritage de la classe ;
— Identifier les classes communes et variables entre deux versions d'un logiciel en se basant sur la déclaration des exceptions de la classe ;
— Identifier les classes communes et variables entre deux versions d'un logiciel en se basant sur les différentes implémentations des méthodes d'une classe donnée ;

Le programme réalisé implémente chacune de ces fonctionnalités et permet à l'utilisateur de choisir quel type de comparaison il souhaite effectuer comme illustré en figure 1.1. Enfin, un exemple d'exécution de comparaison des classes par leur déclaration d'exception est disponible dans le *listing* suivant.

```
Comparaisons will be on this project.Which information do you want?
1. Common and variant classes (same classes name).
2. Common and variant classes (different methods).
3. Common and variant classes (different heritancy).
4. Common and variant classes (different exceptions).
5. Different methods for a given class.
0 To quit.
```

FIGURE 1.1 – Choix entre les différentes méthodes de comparaison.

1. Les versions suivantes se basent également sur le nom de la classe

Listing 1.1 – Résultat de la comparaison en suivant la méthpde de comparaison des classes par leur déclaration d'exception (en plus de leur nom).

```
Following are common classes.
Parser
Main
ClassInfo
MethodBodyLineNumberReverseComparator
ClassAttributeNumberReverseComparator
ClassMethodNumberReverseComparator
MethodParamNumberComparator
MethodInvocationVisitor
FieldDeclarationVisitor
TypeDeclarationVisitor
PackageDeclarationVisitor
MethodDeclarationVisitor
StatsParser
TestSon
AA
BB
CC
Test
CallGraph
Spoon
Couple
Cluster
CouplingParser



Following are different classes.
VariabilityParser
MethodInfo
MethodInf0
```

## 1.2 Exercice 2 - Transformation du code source pour supprimer les dépendances OO

Cette question d'ouverture n'a pas été abordée dans le cadre de notre TP.