TER : Implémentation d’un analyseur de variabilité pour un code configurable annoté.

Contexte

Dans une ligne de produits logiciels, il est possible d’implémenter la variabilité du système à l’intérieur même du code. En utilisant des annotations et un pré-processeur, les développeurs peuvent créer ce que nous appelons un code configurable annoté. Ce code peut rapidement devenir difficile à comprendre plus le nombre d’annotations augmentent. L’analyseur VITAL [Zhang et al. 2014] permet d’offrir des métriques sur ce genre de code, en analysant notamment le nombre d’annotations, leurs tailles (nombre de lignes de code qu’elles entourent) et leurs niveaux d’imbrications (s’il elles sont contenu dans une autre annotation).

Toutefois les analyses s’arrêtent souvent là, les développeurs étant libre d’interpréter ces informations pour effectuer eux-mêmes les changements du code configurables, afin d’en améliorer sa maintenabilité ou compréhension.

Dans ce stage, nous nous intéressons à la notion de qualité d’un code configurable. Nous souhaitons créer un outil permettant d’évaluer la qualité d’un code configurable, et ainsi émettre un score clair sur la qualité du code produit. Cela permettrait aussi de comparer deux codes configurables. De plus, nous souhaitons améliorer les analyseurs existant en récoltant des informations sur le code compris entre les annotations. Cela permettrait d’identifier par exemple les similarités de code entre deux annotations.

Exemple de deux codes annotés qui représente la même variabilité, mais implémenté de deux façons différentes :

```
/*if[RECT]*/
public void wipe() {
	this.rects.clear();
	this.repaint();
}
/*end[RECT]*/

/*if[LINE]*/
public void wipe() {
	this.lines.clear();
	this.repaint();
}
/*end[LINE]*/
```

Points de variations avec une granularité autour des méthodes

```
public void wipe() {
	/*if[LINE]*/
	this.lines.clear();
	/*end[LINE]*/
	/*if[RECT]*/
	this.rects.clear();
	/*end[RECT]*/
	this.repaint();
}
```

Points de variation avec une granularité à l'intérieur des méthodes —> meilleur code configurable

Objectifs :

L'objectif du TER est la conception d'un outil d'analyse, de comparaison et d'évaluation de code configurable de ligne de produits. Pour cela, nous proposons de :

- produire un interpréteur (*parseur*) de code configurable
- identifier ou créer des critères de qualité d'un code configurable
- implémenter ces critères sous la forme de métriques
- analyser le résultat de l'interpréteur vis-à-vis de ses métriques afin d'évaluer la qualité d'un code configurable.
- permettre la comparaison de deux codes configurables vis-à-vis de ces critères.

Bibliographie :

- B. Zhang, « VITAL : reengineering variability specifications and realizations in software product lines », PhD Thesis, University of Kaiserslautern, 2015.
- C. Kästner et S. Apel, « Integrating Compositional and Annotative Approaches for Product Line Engineering », p. 6.
- J. Wilson et T. Ball, *Preprocessing .java with Munge*. 2009.
- J. Meinicke, T. Thüm, R. Schröter, F. Benduhn, T. Leich, et G. Saake, *Mastering Software Variability with FeatureIDE*. Springer, 2017.