DEPARTEMENT INFORMATIQUE
DE LA FACULTE DES SCIENCES

Ahmed Kaci, Guilhèm Blanchard et Yanis Allouch

TER : Synthèse et Analyse de la méthode VITAL et de ses métrique

HMIN201 — Travail d'Etude et de Recherche

Référent: Nicolas Hlad

2021

Avec l'évolution des lignes de produits, des problèmes de maintenance sont apparus, car au fur sont à mesure qu'elles évoluent dans le temps la réalisation de la variabilité deviennent de plus en plus complexes.

L'outil VITAL (Variability ImprovemenT AnaLysis) qui permet d'analyser ce problème, est développé pour extraire automatiquement un modèle de réflexion de variabilité à partir d'un code annoté et mené des analyses complémentaires automatiques. Ainsi, cet outil :

1. Analyse le code de variabilité qui utilise différents mécanismes d'implémentation (par exemple l'algorithme naïf de parsing CPP figure 3 et 4, voir TypeChef pour un outil de parsing plus évolué) et extrait un modèle de réflexion de variabilité dans une base de données (Voir figure 5 : méta modèle de réflexion.).
2. Ensuite, différentes mesures de code de variabilité peuvent être effectuées en exécutant des requêtes de base de données (quelles requêtes ?).
3. En outre, en se basant sur la BDD MS Access, une analyse et une visualisation supplémentaires du code sont effectuées pour pouvoir être utilisé par des outils d'analyse standard tels que MS Excel, R et Treeviz.

Nous présentons les concepts de base sur lesquels se repose la méthode.

— La variabilité (VAR), représente une caractéristique variable dans l'espace du problème, la variable de l'annotation #if.

— Le point de variation (PV) : espace de solutions où est réalisé VAR pour sélectionner les variantes de code (CV).

— Le code variant (CV), le code définit dans un bloc annoté appartenant à un PV.

— Groupe de points de variation (GPV) : ensemble de PV ayant un prédicat logiquement équivalent pour la sélection de fragments de codes.

— Les GPVs seront instanciés par la même caractéristique variable (VAR) avec la même logique.

Voici l'équivalence entre le vocabulaire français et Anglais pour des références futures.

Concept	Anglais	Raccourçi FR	Raccourçi EN
Variabilité	Variability	VAR	VAR
Point de Variation	Variation Point	PV	VP
Code Variant	Code Variant	CV	CV
Groupe de Points de Variation	Variation Point Group	GPV	VPG

Nous avons pu extraire les métriques de l'article suivant [ZB14].

— VP Nesting Degree ➔ Degré/Niveau d'imbrication d'un Point de Variation. (V.P. <=> P.V.)

— Var Tangling Degree ➔ Plus une variable est utilisé dans un VP, plus elle est emmêlée dans le code (à éviter ?)

— Var Fan-out on VPG ➔ On s'intéresse à la présence d'une variable (ou bien de l'ensemble des variables ?) dans un VPG et VITAL fait la moyenne de la dispersion, de la présence, de la répartition a travers le VPG.
➔ autre façon de voir ?

Var Fan-out on VPG ➔ On s'intéresse à la présence d'une variable (ou bien de l'ensemble des variables ?) dans un VPG et VITAL fait la moyenne du nombre de variable apparu par apport au nombre de VP du VPG.

— Var Fan-out on File ➔ On s'intéresse à la même métrique que ci dessus mais sur un fichier.

— Var Fan-in on File ➔ On répertorie le nombre de Vars dans un fichier (le faire sur tous les fichiers ?)

— VP Fan-in on File ➔ On compte le nombre de VP dans un fichier donné et on sauvegarde dans la BDD pour pouvoir faire des rquêtes. (Deux VP équivalents dans un même fichier feraient augmenté sa présence ?)

— Le nombre fichiers possédants des points de variations.

— Nombre de ligne (LOC) de code annoté.

Question pour N. Hlad

— Qu'est ce qu'un modèle de réflexion ?

— Est ce que les macros telles que #if, #ifndef et #elif fonctionne aussi sur FeatureIDE ?

— Dans un code **filtrer** peut-on proposer l'outil de transformation de code source en code filtré avec une syntaxe suivante

```
#if
    // @LOC 7
    // @paramXYZ valueFooBar
        #if X > 10
            // @func true
            // @funcArgsNumber 5
            // @md5 @line X == d465fg44s654465465496874fdg9684h65
            // @md5 @line X+1 == 4984q324d65tujh796584796eryt7i98uk
            // (par exemple pour faire la metrique correspondance de code entre deux VP)
            // imaginez une syntaxe/structure/standard pour analyser la metrique sur du code filtre.
        #endif
#endif

```

— Problème de collusion avec le md5 ? de sécurité ? voir d'autre protocole.

— Quels convention de nommage est utilisé par isiSPL ? Y 'a t-il une spécification pour la génération de nom de feature ?

— Si le tangling diminue est ce que cela veut dire que le nombre de variable diminue (?) Dans ce cas est ce que le code est plus facile à lire/prendre en main, ou alors le code possède toujours autant de variable mais moins éparpillé et emmêlée entre elle et donc même conclusion ?.

— Peut on appliquer le tangling aux VPG ? une VPG = ensemble de VP Équivalent qui contient déjà la Var ?

Idées

— B. Zhang introduit un outil plus évolué pour parser du code source : TypeChef. Une rapide recherche nous fait penser qu'on peut récupérer les recommandations de leur analyse de type checking variabilité comme métriques (parce qu'un code faux, est un mauvais code) ?

Références

[ZB14] Zhang, B. et Becker, M. « Variability code analysis using the VITAL tool ». In : *Proceedings of the 6th International Workshop on Feature-Oriented Software Development - FOSD '14*. ACM Press, 2014.