ÉLÉMENT DE PORTFOLIO 04

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : A novel approach for Software Architecture Product Line Engineering. Publié dans la revue Journal of Systems and Software, vol. 186, pp. 111191, (Elsevier) (2022)

URL de l'élément : `https://hal.science/hal-03885616`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Il s'agit d'un article publié dans la revue Journal of Systems and Software, une revue de rang A qui est une extension d'un premier article publié dans la conférence ICECCS 2019 [2]. Nous avons choisi de présenter cet article car il montre l'impact très positif de nos développements logiciels sur notre visibilité. L'article présente une extension de l'outil BUT4Reuse pour les architectures logicielles.

3 PRÉSENTATION DE CET ÉLÉMENT

3.1 Contexte

BUT4Reuse (Bottom-Up Technology for Reuse) est le logiciel développé au sein de l'équipe MoVe pour l'extraction (ou l'inférence) de lignes de produits logiciels à partir d'artefacts logiciels existants. BUT4Reuse a été conçu comme un cadre générique et extensible pour supporter différent types d'artefacts (modèles, code source, etc.) [3]. Dans cet article, nous avons considéré l'extraction de lignes de produits à partir d'artefacts logiciels représentant des architectures logicielles. Nous avons considéré des architectures logicielles obtenues à partir de code source de systèmes développés dans le cadre de l'OSGi, tels que l'IDE Eclipse. La spécification OSGi définit un modèle de composant et un cadre de travail pour créer des systèmes Java hautement modulaires [4]. Les IDE basés sur Eclipse sont une collection de produits logiciels similaires qui partagent un ensemble d'artefacts logiciels. Extraire une ligne de produits à partir de cette collection représente l'objectif principal traité dans cet article.

Dans ce contexte, nous avons identifié cinq défis principaux : i) Comment extraire une architecture logicielle à partir de code source de chaque variante ; ii) Comment comparer les variantes d'architecture logicielle pour identifier les parties communes et trouver les caractéristiques (features) variables ; iii) Comment construire la Ligne de Produits Architecturale (LdPA) avec une spécification explicite de la variabilité au niveau architectural ; iv) Comment simplifier et réduire la complexité des architectures récupérées. Cet article propose une approche permettant de résoudre tous ces défis. Une validation expérimentale intensive a été présentée en utilisant le benchmark EFLBench [4].

3.2 Résultats

L'approche proposée est composée de deux processus principaux : i) un processus ascendant (P1) d'inférences de LdPA ; ce sous-processus se commence d'abord par la rétro-ingénierie des architectures logicielles à partir du code source de chaque variante logicielle. Ensuite, il reconstruit un LdPA pour ces variantes d'architecture logicielle ; t ii) un processus de dérivation (P2) qui permet de dériver de nouvelles variantes (variantes d'architecture logicielle).

Inférence de LdPA. La construction de LdPA à partir de variantes d'architecture logicielle est réalisée en implémentant un nouvel adaptateur pour BUT4Reuse. Cet adaptateur suit les principes généraux de BUT4Reuse en définissant : 1) les éléments atomiques ; 2) la fonction de similarité ; 3) la définition des contraintes structurelles. Les algorithmes d'identification de blocs de BUT4Reuse sont utilisés par la suite pour la comparaison et l'inférence de la LdPA. La LdPA obtenue est composée d'une part, d'un feature model spécifiant la variabilité et d'autre part, de l'ensemble de fragments d'architecture associés à chacune des features selon une approche compositionnelle [1].

Dérivation de variantes d'architecture logicielles. Le deuxième processus a comme objectif de créer de nouvelles variantes d'architectures à partir de la LdPA obtenue. Il suit les grandes lignes de dérivation dans les approches compositionnelle en intégrant ce qui est appelé un "composer" qui permet d'assembler les fragments d'architecture en fonction des features sélectionnées.

Expérimentation et validation. Nous avons validé l'approche proposée en utilisant un ensemble de données générées à partir de EFLBench. En effet, EFLBench permet de générer d'une manière intensive des distributions de l'IDE Eclipse. Nous avons identifié deux questions de recherche liées aux deux processus P1 et P2 enrichis avec un ensemble de métriques d'évaluation. Dans nos expérimentations, nous avons généré 100 variantes d'IDE Elipse et les résultats obtenus montrent des taux élevés de précisions . Nous avons utilisé la LdPA obtenue pour générer des variantes qui sont fonctionnellement correctes.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Sven Apel, Don S. Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines - Concepts and Implementation*. Springer, 2013.

[2] Mohamed Lamine Kerdoudi, Tewfik Ziadi, Chouki Tibermacine, and Salah Sadou. Recovering Software Architecture Product Lines. In *ICECCS 2019 - 24th International Conference on Engineering of Complex Computer Systems*, pages 226–235, Nansha, Guangzhou, China, November 2019. IEEE.

[3] Jabier Martinez, Tewfik Ziadi, Tegawendé Bissyandé, Jacques Klein, and Yves Le Traon. Bottom-Up Adoption of Software Product Lines - A Generic and Extensible Approach. In *19th International Software Product Line Conference (SPLC)*, pages 101–110, Nashville, TN, United States, July 2015. ACM.

[4] Jabier F Martinez, Tewfik Ziadi, Mike Papadakis, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Feature location benchmark for extractive software product line adoption research using realistic and synthetic Eclipse variants. *Information and Software Technology*, July 2018.