Processus pour la génération automatique de composants exécutables à partir de contraintes d'architecture

Sahar Kallel*,**, Bastien Tramoni*,
Chouki Tibermacine*, Christophe Dony* et Ahmed Hadj Kacem**

*LIRMM, CNRS et Université de Montpellier, France
sahar.kallel, chouki.tibermacine, bastien.tramoni, dony@lirmm.fr,
**ReDCAD, Université de Sfax, Tunisie
sahar.kallel@redcad.org, ahmed.hadjkacem@fsegs.rnu.tn

Résumé. Les contraintes d'architecture sont des spécifications définies par les développeurs dans la phase de conception, qui permettent de vérifier, après une évolution de l'architecture, si sa description est encore conforme aux conditions imposées par un patron ou un style architectural, ou bien une règle de conception générale. Ces spécifications peuvent être exprimées avec un langage standardisé comme OCL. Elles sont la plupart du temps des spécifications brutes sans aucune structure permettant leur paramétrage et réutilisation. Afin de pouvoir les vérifier dans la phase d'implémentation nous proposons dans ce travail une méthode pour traduire automatiquement ces spécifications en composants exécutables. En plus de les rendre vérifiables en phase d'implémentation, nous avons choisi de cibler les composants logiciels afin de rendre ces contraintes d'architecture réutilisables, personnalisables et composables. Puisque les contraintes d'architecture doivent analyser les descriptions d'architecture, les composants générés utilisent le mécanisme de réflexivité standard fourni par le langage de programmation. Notre implémentation prend en entrée des contraintes OCL spécifiées sur le métamodèle UML. Elle produit en sortie des composants programmés en COMPO, un langage de programmation par composants réflexif développé par notre équipe.

1 Introduction : Contexte et Problématique

Les contraintes d'architecture sont des spécifications d'invariants qui sont vérifiables par l'analyse des descriptions d'architecture. Ce genre de contraintes ne doit pas être confondu avec les contraintes fonctionnelles, qui sont vérifiables par l'analyse de l'état des composants exécutables constituant l'architecture. Par exemple, si on considère un modèle UML (une description d'architecture) contenant une classe `Employé` (un composant dans cette architecture) qui a un attribut `age`, une contrainte fonctionnelle représentant un invariant sur cette classe peut tester les valeurs de cet attribut pour qu'elles soient toujours comprises dans l'intervalle 16 à 70. Cette contrainte sera vérifiée sur toutes les instances de la classe *Employé*. Ce genre de contraintes est intrinsèquement dynamique. Elles ne peuvent être vérifiées que lors de l'exécution.