Transformation de programme à l'aide de la programmation par aspects

Soutenance intermédiaire
Rotovei Doru

1 Introduction

Depuis quelques années les modèles de composants sont apparus dans le monde de la conception logiciels pour résoudre les problèmes de répartition, de déploiement et de réutilisation de code.L'objectif du stage est de présenter les transformations de modèles de composants à l'aide de la programmation par aspects dynamiques. Cette étude s'inscrit dans la problématique de transformation de modèle de l'approche MDA(Model Driven Architecture). Il y a trois an l'OMG (Object Management Group) a introdui une nouvelle orientation pour résoudre les problèmes actuels de construction et d'évolution de systèmes d'information. Le nom de cette nouvelle orientation est Model Driven Architecture (MDA) ou Architecture dirigée par les modèles [7]. Le principe de base du MDA est l'élaboration de modèles indépendants de plates - formes (PIM) - donc des modèles de plus haut niveau d'abstraction - et la transformation de ceux-ci en modèles dépendants de plates-formes (PSM). Cependant, le MDA ne donne aucune règle de transformation du PIM vers PSM. Une règle universelle n'existe pas encore mais beaucoup de projets de recherche travaillent pour trouver les transformations les plus génériques possible. Dans le cadre du mon sujet de stage, nous essayons, aussi, de proposer une méthode de transformation du PIM vers PSM à l'aide d'aspects dynamiques.

La programmation par aspects (AOP)[1] se fonde sur une séparation claire entre les préoccupations *"fonctionnelles"* et *"non-fonctionnelles"* présentes dans les applications. Chaque aspect est destiné à être développé de façon indépendante puis intègre à une application par un processus de tissage d'aspects (aspect weaving).

Il existe actuellement plusieurs implémentations des aspects pour divers langages à Objet comme pour C++ ou Java, mais ces implémentations ont comme caractéristique commune d'appliquer les mécanismes de transformation du programme source, et donc par nature, suivent une approche plutôt statique pour l'implémentation des aspects. Dans le cadre de *SmartTools* [2] une forme particulière d'aspect a été développe qui a comme caractéristique commune l'aspect dynamique, spécifiques a nos analyses sémantiques basées sur le patron de conception visiteur[6]. Le branchement, comme le débranchement d'un aspect, peut se faire dynamiquement et à tout moment pendant l'exécution du programme[5]. Cependant, les point de jonction dans le cas de SmartTools sont limités juste avant et après une méthode *visit*. Le patron de conception visiteur permet de définir de nouvelles opérations sur une structure d'objets en séparant le code (le visiteur) et la structure. La structure, en ce cas, correspond à l'arbre de syntaxe abstraite du composant, pendant que les visiteurs forment, à l'aide d'aspects, la partie fonctionnelle du composant. Les composants apportent ses propriétés d'autonomie, d'auto-description de réutilisation et de dynamicité, et les aspects apportent la sémantique et la fonctionnalité. La fonctionnalité et la sémantique seront projetés, à l'aide d'aspects dynamiques, vers différents technologies de composants comme EJB CORBA ou Web Services. Comme chaque aspect peut décrire une fonction du composant, il faut faire un lien entre la notion de composant et ses aspects fonctionnels. Le problème est d'obtenir un composant suffisamment générique pour pouvoir ajouter ses parties sémantiques juste avec des aspects dynamiques. Il existe actuellement, plusieurs technologies de composants et pour ça une nécessité de construire des modèles indépendant de plate - forme – qui ne tiennent pas compte du caractéristiques techniques – et de projeter ceux-ci vers n'importe quel modèle spécifique de composant tel que CORBA,Web Services,Entreprise Java Beans etc. À partir d'un modèle abstraite de composant (abstraction au sens d'une réduction des détails non significatifs), nous pouvons construire des aspects qui pourront faire la projection vers une certaine plate-forme techniques de composant. Les aspects

seront ajouter aux visiteurs qui parcourent l'arbre de syntaxe abstraite correspondant au modèle de composants. À partir d'un autre modèle de base, en utilisant les mêmes aspects, mais en changeant les points de jonctions au visiteurs, nous pouvons obtenir une projection du ce modèle vers la même plate - forme techniques. Donc l'idée de base du cette étude est de trouver les aspects qui permettront faire la projection, et aussi de construire les aspects indépendant du modèle de base pour pouvoir les réutiliser pour n'importe quel autre modèle.

2 État de l'art

Les recherches et les méthodologies traitant les transformations du PIM vers PSM sont encore au stade préliminaire ou en cours d'être construites. Cependant, beaucoup des propositions sont faits et j'essayerai citer quelques un.

Dans l'article [3], les auteurs proposent une approche par vues pour la réalisation d'aspects fonctionnels. Ils considèrent les approches composants et vues dans un même modèle. Le composant apporte ses propriétés de réutilisation et configuration et les vues apportent une méthode de structuration cohérent et plus riche que l'approche objet traditionnelle. Ils travaillent aussi sur la transformation du leur modèle abstraite vers de modèle spécifiques, mais la plus grande partie de la génération de code se réalise a partir du PSM (les modèles spécifiques) et pas directement depuis leur modèle abstraite.

L'outil GenVoca [4] présente quelques formalismes pour générer des composants et des langages métier pour le langage Java en l'étendant avec de nouvelles fonctionnalités . L'accent se mis surtout sur les interactions entre les outils qui doivent être compatibles avec GenVoca et qui peuvent être très bien utilisés indépendamment, que sur la génération des composants. En effet, chaque outil peut être considéré comme un composant indépendant d'autres. Les auteurs insistent bien sur les formalismes envisageant les interactions entre les outils, qui se font en respectant les spécifications du générateur GenVoca.

La plate-forme PROSE[8] propose un environnement qui support les aspects dynamiques, construit au dessus du langage Java. Le but principale de la plate-forme PROSE est d'avoir un environnement pour le dépannage et le prototypage pour le tissage d'aspects (aspect weaving).

Cependant, actuellement une approche unificatrice entre MDA, AOP e composant n'existe pas. Les transformations du modèles abstraits de composants vers de modèles spécifiques, en utilisant les aspects dynamiques ainsi que la problématique de transformation de modèle de l'approche MDA, sont l'objectif du ce stage.

Références

[1] Aspect-Oriented Programming. http://www.parc.xerox.com/csl/projects/aop/.

[2] SmartTools home page. http://www-sop.inria.fr/oasis/SmartTools/.

[3] Gilles Vanwormhoudt Alexis Muller, Oliver Caron Bernard Carré. Réutilisation d'aspects fonctionnels : des vues aux composants. In *Langages et Modèles à Objets*, volume 9 of *L'objet*, pages 241–255, 2003.

[4] Don Batory, Bernie Lofaso, and Yannis Smaragdakis. JTS : tools for implementing domain-specific languages. In *Proceedings Fifth International Conference on Software Reuse*, pages 143–153, Victoria, BC, Canada, 2–5 1998. IEEE.

[5] Carine Courbis. *Contribution à la programmation générative. Application dans le générateur* SMARTTOOLS *: technologies XML, programmation par aspects et composants*. PhD thesis, Université de Nice Sophia-Antipolis, INRIA Sophia-Antipolis, décembre 2002. http://www-sop.inria.fr/oasis/personnel/Carine.Courbis/these.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, Reading, MA, 1995. ISBN 0-201-63361-2-(3).

[7] OMG. MDA - Model-Driven Architecture. http://www.omg.org/mda.

[8] Andrei Popovici, Thomas Gross, and Gustavo Alonso. Dynamic weaving for aspect-oriented programming. In *Proceedings of the 1st international conference on Aspect-oriented software development*, pages 141–147. ACM Press, 2002.