COLOSS et OBASCO

Sujet de thèse 2007

Programmation par aspects et composants

Encadrants : Gilles Ardourel et Rémi Douence (douence@emn.fr)
Équipe d'accueil : COLOSS (Université de Nantes, Lina) et OBASCO (École des Mines de Nantes-Inria, Lina)
Mots-clés : programmation par aspects, composants logiciels, architectures logicielles

Description du sujet

On assiste actuellement à l'émergence de deux formes de programmation visant à améliorer la réutilisation et l'adaptabilité du logiciel au travers d'une meilleure séparation des préoccupations tout au long du développement d'une application : la programmation par composants et la programmation par aspects [1, 2, 3].

Il s'agit dans les deux cas de développer des applications par assemblage, de composants dans le premier cas et d'aspects dans le deuxième cas, avec les mêmes problématiques (vérifier la correction des assemblages, autoriser les assemblages dynamiques...). Cet assemblage prend toutefois des formes différentes. Il est symétrique dans le cas des composants, chaque composant fournissant explicitement des services en s'appuyant sur des services fournis par d'autres composants. Il est asymétrique dans le cas des aspects, avec la définition d'un aspect de base correspondant à une structure et un comportement principal, sur lequel viennent se greffer de manière transparente (sans référence explicite dans le programme) des aspects additionnels traitant des préoccupations non couvertes par l'aspect de base.

Ces deux formes d'assemblage s'avèrent complémentaires. On peut ainsi avoir envie de structurer un aspect complexe sous la forme d'un assemblage de composants et un composant comme un assemblage d'aspects. Ce dernier point de vue point est notamment naturel dans le cadre de l'assemblage de composants comme les Enterprise JavaBeans [4] ou les services web [5]. Le code métier du composant constitue alors l'aspect de base, complété par des aspects techniques gérant des préoccupations comme la distribution, la sécurité, la persistance, sans référence explicite à ces notions dans le code métier. Cette complémentarité est maintenant reconnue et a conduit à la définition de cadres de programmation en Java comme JBoss AOP [6] ou Spring [7]. Ce type d'intégration reste toutefois lâche, peu expressif et difficile à maîtriser.

Il s'agira donc d'étudier, à partir de l'expérience des équipe Coloss et Obasco dans le domaine des langages d'aspects et des langages de composants, la possibilité d'une intégration plus fine des composants et des aspects sous la forme d'un langage de programmation unifié. Nous nous intéresserons à la fois à la définition bien fondée d'un tel langage, à sa mise en œuvre et à son utilisation pratique. Le langage ou l'environnement devront supporter l'expression et la vérification de propriétés ou de contraintes portant sur l'assemblage et les aspects. Par exemple on veut pouvoir garantir la conformité des interactions entre deux composants après ajout d'un aspect, ou un invariant sur l'état d'un composant. Une première piste à explorer concerne la notion de vue. Dans un système hiérarchique on peut distinguer deux types de composants : les composants primitifs (c.a.d. les boîtes) et les composants composites (c.a.d. les boîtes de boîtes). Un unique système défini par ses composants primitifs peut être structuré de multiples façons à l'aide de composants composites ou de vues. Une vue peut être considérée comme un composite représentant une préoccupation ou un domaine dont l'enveloppe est calculée et qui permet d'intercepter les communications d'un groupe de composants à l'aide d'un aspect. Se posent alors les questions de la cohabitation de plusieurs vues d'un même système (contrôle, cohérence, coût) et du recalcul des vues lors d'une reconfiguration dynamique du système.

Ce travail s'effectuera dans le cadre du projet régional Miles. Il correspond au démarrage d'une nouvelle étape dans les travaux communs aux deux équipes.

Références

[1] Clemens Szyperski. *Component Software*. Addison-Wesley, 2002. 2nd edition.

[2] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Aksit and S. Matsuoka, editors, *ECOOP'97 - Object-Oriented Programming - 11th European Conference*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242, Jyväskylä, Finnland, June 1997. Springer-Verlag.

[3] Jacques Noyé, Rémi Douence, and Mario Sudhölt. Composants et aspects. In Mourrad Oussalah, editor, *Composants : Concepts, techniques et outils*, chapter 6. Vuibert, 2005.

[4] L.G. DeMichiel. *Enterprise JavaBeans^{TM} Specification*. SUN Microsystems, November 2003. Version 2.1, Final Release.

[5] B. Verheecke, M. A. Cibrán, W. Vanderperren, D. Suvée, and V. Jonckers. AOP for dynamic configuration and management of web services in client-applications. *International Journal on Web Services Research (JWSR)*, 1(3), July-September 2004.

[6] JBoss. The JBoss AOP website. `http ://www.jboss.org/developers/projects/jboss/aop`, 2005.

[7] Spring. The Spring website. `http ://www.springframework.org`, 2005.