Sujet de master recherche « Architectures logicielles distribuées » 2005–2006

KAO[1] : Un langage de composants et d'aspects intégrés

Encadrant principal : Jacques Noyé
courriel : `Jacques.Noye@emn.fr`

Objectif du stage

Jusqu'à présent, la programmation par composants et la programmation par aspects se sont développées chacune de leur côté. Pourtant, tant les composants que les aspects sont supposés aider à la séparation des préoccupations tout au long du développement d'une application. En fait, le discours dominant est que les composants correspondent à des préoccupations de *base* (on parle aussi de préoccupation *métier* ou *fonctionnelle*) alors que les aspects correspondent à des préoccupations *transverses* (on parle aussi de préoccupations *techniques* ou *non fonctionnelles*). En pratique, cette différence peut toutefois s'avérer fort subtile.

Une manière de rapprocher composants et aspects et de mieux apprécier leurs similitudes et leurs différences est de considérer la notion de collaboration [1]. Cette notion appartient au folklore de la conception et de l'analyse par objets. Une collaboration peut être vue comme une manière de capturer une préoccupation dans une application à base d'objets. Elle est définie par un ensemble d'objets et un *protocole* expliquant comment ces objets interagissent pour implémenter une préoccupation donnée. Le protocole definit pour chaque objet le role de l'objet dans la collaboration. Bien sûr, un objet peut appartenir à plusieurs collaborations. Plusieurs langages ont récemment été développés autour de cette idée, avec des objectifs différents. Ainsi, Caesar [2] est un langage d'aspects, JL [3] et Jiazzi [4] sont des langages de composants, le langage des *AspectualCollaborations* [5] est un langage unifiant modules et aspects. Mais aucun de ces langages n'intègre véritablement composants et aspects. En particulier, les collaborations aspectuelles n'ont pas d'existence à l'exécution.

Travail à réaliser

Un métamodèle des langages d'aspects, combinant des techniques des langages de programmation et des techniques d'ingénierie des modèles, est en cours de définition au sein du réseau d'excellence européen AOSD (*Aspect-Oriented Software Development* [6]). On s'intéressera à la prise en compte par ce métamodèle de la notion de composant afin de définir un langage de programmation d'architecture permettant la reconfiguration dynamique de composants et d'aspects et son interprète. Si le temps le permet, on s'intéressera aussi à la possibilité de générer, à partir de ce métamodèle, du code efficace, par exemple vers l'infrastructure de programmation par aspects Reflex [7, 8].

Références

[1] Jacques Noyé, Rémi Douence, and Mario Sudhölt. Composants et aspects. In Mourrad Oussalah, editor, *Ingénierie des composants : Concepts, techniques et outils*, chapter 6, pages 169–195. Vuibert, February 2005.

[1] Ici, pour Komponent (c'est de l'allemand, ou du danois, du suédois...), Aspect et Objet. Kao veut aussi dire « haut » en chinois.

[2] Mira Mezini and Klaus Ostermann. Conquering aspects with Caesar. In Mehmet Aksit, editor, *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD 2003)*, pages 90–99, Boston, Massachusetts, USA, March 2003. ACM Press.

[3] R. Cardone and C. Lin. Comparing frameworks and layered refinement. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 285–294, Toronto, Canada, May 2001. IEEE Computer Society Press.

[4] Sean McDirmid, Matthew Flatt, and Wilson C. Hsieh. Jiazzi : New-age components for old-fashioned Java. In *Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications (OOPSLA'01)*, pages 211–222, Tampa Bay, FL, USA, October 2001. ACM Press.

[5] David H. Lorenz, Karl Lieberherr, and Johan Ovlinger. Aspectual collaborations : Combining modules and aspects. *The Computer Journal*, 46(5) :542–565, September 2003.

[6] AOSD Europe. The aosd europe web site. http ://www.aosd-europe.net/.

[7] Éric Tanter, Jacques Noyé, Denis Caromel, and Pierre Cointe. Partial behavioral reflection : Spatial and temporal selection of reification. In R. Crocker and G.L. Steele Jr., editors, *OOPSLA 2003, Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 27–46, Anaheim, CA, USA, October 2003. ACM Press. ACM SIGPLAN Notices, 38(11).

[8] Éric Tanter and Jacques Noyé. A versatile kernel for multi-language AOP. In Robert Glück and Michael Lowry, editors, *Proceedings of the 4th International Conference on Generative Programming and Component Engineering (GPCE'05)*, volume 3676 of *Lecture Notes in Computer Science*, pages 173–188, Tallinn, Estonia, September/October 2005. Springer-Verlag.