l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

ECOLE DES MINES DE NANTES

THESE

Abdelhakim Hannousse

ECOLE DOCTORALE : ED STIM
THESE N° 2011EMNA0009

Thèse présentée en vue de l'obtention du grade de
Docteur de l'Ecole des Mines
Sous le label de l'Université Nantes Angers Le Mans
Discipline ***Informatique***

Soutenue le 14 novembre 2011

Aspectualizing Component Models : Implementation and Interferences Analysis

DIRECTEUR DE THESE **:**
Mario Südholt, Professeur, EMN
CO DIRECTEUR DE THESE :
Rémi Douence, Maître de conférence, INRIA, EMN
Gilles Ardourel, Maître de conférence, Université de Nantes

RAPPORTEURS DE THESE :
Laurence Duchien, Professeur, Université de Lile 1 (USTL)
Isabelle Borne, Professeur, Université de Bretagne Sud

PRESIDENT DU JURY :
Jean-Marc Jézéquel, Professeur, Université de Rennes

MEMBRES DU JURY :
Laurence Duchien, Professeur, Université de Lile 1 (USTL)
Isabelle Borne, Professeur, Université de Bretagne Sud
Mario Südholt, Professeur, EMN
Rémi Douence, Maître de conférence, INRIA, EMN
Gilles Ardourel, Maître de conférence, Université de Nantes

Acknowledgements

The work presented in this thesis could not have been possible without the support of many people. Many thanks to my supervisors: Mario Südholt, Rémi Douence and Gilles Ardourel for their timely advice, consultations, encouragement, and critiques throughout the development of this work. Rémi has been an invaluable source of support and guidance all along my work on the thesis.

Many thanks to my small family (my wife and my daughter Rawane), I also would like to thank my Mom and Dad for their encouragement, understanding and caring. They have always been a source of motivation. Without them, I could have never accomplished what I have done today.

Last and not least, I am very grateful to Ascola and Aelos teams for their support of this thesis.

Abdelhakim Hannousse

Abstract

Component based software engineering, or CBSE in short, enables the modularization of concerns in terms of separate software entities called components. Each component provides a set of services and may require services from other components to accomplish its tasks. Components can be assembled in order to construct complex systems. On the other hand, aspect oriented programming, or AOP in short, focuses on the modularization of scattered and tangled concerns that cannot be modularized using regular software entities. Crosscutting concerns are not related to a specific paradigm and CBSE is not an exception. However, current works on CBSE focus only on mapping AspectJ-like concepts into component models missing the particularity of components (*i.e.,* join point model for black boxes) and component systems (*i.e.,* pointcuts defining points in component architectures) and the interferences that may appear when several aspects are woven to a system. In fact, aspect interferences detection and resolution is still a challenge for AOP. In this thesis we contribute by introducing a declarative pointcut language (VIL) for component models, and we provide a formal framework for aspect interference detection and resolution when several aspects are woven to a component system. In our framework we introduce an ADL that extends current ADL(s) with explicit definition of component and aspect behaviors, and aspect weaving rules. Each weaving rule uses our VIL expressions to describe which and where aspects should be woven. We provide a set of transformation rules to obtain the formal specification of components and aspects from the ADL, and we use model checkers for the detection of potential interferences among aspects. For interference resolution, we provide a set of composition operators. Each operator is given with a motivation example, a structure, and a set of applicability rules. The operator structure is described as an abstract form that can be instantiated for any two arbitrary aspects and a set of join points. The list of operators in the catalog is not exhaustive but it can be considered as a first step towards a pattern catalog for aspect interferences resolution. In our framework we adopt the use of Uppaal model checker for its support of template instantiation, local variable declarations, and parameter passing between processes in addition to its support of timing constraints to model real time systems. We illustrate our approach with Fractal component model and a case study: airport wireless access. We define a set of interfering aspects for the example, and we show how our modelization of the system with aspects in Uppaal enables the detection of interferences and how our operators can be instantiated to solve them. Finally, we should mention that our framework is general and can be used for other component models with minimum adaptations.

Résumé

La programmation par composants (CBSE) permet la modularisation des préoccupations en termes d'entités logiciels séparées appelés composants. Chaque composant fournissant explicitement des services en s'appuyant sur des services fournis par d'autres composants. Les composants peuvent être assemblés afin de construire le système global. D'autre part, l'approche aspects (AOP) vise à séparer les préoccupations techniques ou de contrôle (*e.g.,* synchronisation, persistance, contraintes temps réel, etc.) des préoccupations métier ou fonctionnelles. Elle offre un mécanisme de tissage qui permet de fusionner ces deux types de préoccupations afin de construire le système global. Ceci permet une meilleure séparation du code fonctionnel du code non fonctionnel et d'assurer une meilleure maintenabilité du système. Les préoccupations transversales ne sont pas liés à un paradigme spécifique et le paradigme composants n'est pas une exception. Malheureusement, les travaux actuels sur la programmation par composants vise à implenté les concepts d'AspectJ directemment on tel quels dans les modéles à composants ignorant la particularité des composants et les systémes à composants (*i.e.,* points de coupures définissants des points dans les architectures composants) et les interférences entre les aspects qui peuvent apparaissent lorsque plusieurs aspects sont tissés à un systéme. En fait, la détection et la résolution des interférences d'aspects est toujours un défi pour les AOP. Dans cette thése, nous contribuons par l'introduction d'un langage déclarative de points de coupure (VIL) dédié au modèles à composants, et nous fournissent un cadre formel pour la détection et la résolution des interférences d'aspects lorsque plusieurs aspects sont tissés à un système à composants. Dans ce cadre, nous introduisons un ADL qui s'étend ADL(s) actuellent par une définition explicite des comportements des composants et d'aspects, et les règles du tissage et de composition d'aspects. Chaque règle utilise des expressions VIL afin de décrire déclarativement où les aspects vont être tissés. Nous fournissons un ensemble de règles de transformation pour obtenir la spécification formelle des composants et des aspects à partir de l'ADL, et nous utilisons des model checkers pour la détection des interférences possibles entre les aspects. Pour la résolution des interférences, nous fournissons un ensemble d'opérateurs de composition. Chaque opérateur est donnée avec un exemple de motivation, une structure et un ensemble de règles d'applicabilité. La structure d'opérateur est décrit comme une forme abstraite qui peut être instancié pour n'importe quel deux aspects et n'importe quelle ensemble de points de jonture. La liste des opérateurs forme une première étape vers un catalogue pour la résolution d'interférences d'aspects. Dans notre proposition, nous adoptons l'utilisation de model checker Uppaal pour son soutien à l'instanciation des processus, la déclaration des variables locales, et le passage de paramètres entre les processus, en plus de son soutien à des contraintes temporelles pour modéliser les systèmes temps réel. Nous illustrons notre approche avec le modèle de composants Fractal et une étude de cas: l'accès wifi dans un AirPort. Nous définissons un ensemble d'aspects interférant pour l'exemple, et nous montrons comment notre modélisation du système avec les aspects en Uppaal permet la détection d'interférences et de la façon

dont nos opérateurs peuvent être instanciés pour les résoudre. Enfin, il convient de mentionner que notre approche est générale et peut être utilisé pour d'autres modèles à composants avec des adaptations minimales.

Contents

5 Aspects as wrappers on views of component systems architectures 91
5.1 Aspects as wrappers on views . . . 91
5.2 Views definition language . . . 96
5.2.1 The join point Model . . . 96
5.2.2 Syntax of VIL . . . 97
5.2.3 Semantics of VIL . . . 98
5.2.3.1 FPath Query Language . . . 98
5.2.3.2 VIL semantics in FPath . . . 99
5.3 Implementation of VIL in Fractal component model . . . 101
5.3.1 Composable controllers . . . 101
5.3.2 The components of interest belong to the same composite . . 103
5.3.3 The components of interest are scattered in the architecture . 106
5.3.4 Fractal Weaver . . . 109
5.3.4.1 VIL Analyzer . . . 109
5.3.4.2 ADL Transformer . . . 110
5.3.4.3 Julia Config Generator . . . 110