Language facilities for the deployment of reusable aspects

Bruno De Fraine

In the domain of aspect-oriented software development, there is a high level of interest in reusable aspect modules by both researchers and practitioners. A number of approaches target the incorporation of aspect mechanisms in a component-based style of software development. But also in the context of traditional aspect systems the concept of aspect libraries with reusable implementations of crosscutting concerns is becoming increasingly relevant.

At the level of the aspect programming language, considerable attention has been devoted to powerful abstraction mechanisms, to be able to describe aspect behavior independent of a specific context, with flexible extension points. However, a separate deployment step is also needed, to configure and activate reusable aspect behavior for a concrete setting. We claim that advanced deployment mechanisms are equally important to enable the intensive use and reuse of aspects. We consider two dimensions of support for aspect deployment:

- *Expressive* deployment mechanisms allow using existing aspects in more contexts. They improve the general software engineering properties of deployment logic.
- *Safe* deployment mechanisms automatically verify the compatibility between the aspect and the context where it is used. This makes the reuse of aspects more manageable.

We find that the deployment mechanisms of current aspect approaches are lacking in these two respects, and present contributions in each area.

With respect to the expressiveness of deployment mechanisms, we analyze the current mainstream deployment constructs and identify a number of common shortcomings that relate to (i) reuse of the deployment logic, (ii) quantification of aspect behavior, and (iii) activation of new deployments at dynamic program events. We propose a language design to remedy these problems and we realize this design as the EcoSys AOP framework. EcoSys employs general program code for the specification of deployment logic, which provides rich abstraction mechanisms and control structures. It also allows for the deployment to be better integrated with the rest of the program.

For the safety of deployments, we focus on the static typing of deployment code. Compared to a full specification and verification of the program behavior, types offer a practical and lightweight approximation of the semantics. Our contribution is the development of flexible typing principles for the safe application of aspect behavior at concrete program points, based on the subtype relations that provide both an upper and lower bound for types, and the use of type variables to support genericity. We demonstrate how the type system can be integrated with the dynamic deployment mechanisms of EcoSys, thanks to generic typing features of recent languages such as Java 5. The proposed typed

EcoSys approach achieves a combination of both expressive and typed aspect deployments, and demonstrates that these two properties should not be considered at odds with each other. In addition, we present the StrongAspectJ language, which integrates the proposed typing principles in a mainstream aspect language. Finally, we provide a formal evaluation of the proposed typing principles with a rigorous proof of the safety properties in the context of the Featherweight Java calculus.