Language Oriented Modularity: From Theory to Practice *

Arik Hadas

Open University of Israel
Raanana 43107, Israel
arik.hadas@openu.ac.il

Abstract

Language Oriented Modularity (LOM) is a methodology that is based on Language Oriented Programming (LOP) and applied to Domain-Specific Aspect Languages (DSALs) rather than Domain-Specific Languages (DSLs). Like LOP, which involves the development and use of DSLs on-demand during the software development process, LOM involves the development and use of DSALs on-demand during the software modularization process. However, LOM is underutilized and often not used at all in modern projects. The goal of this research is to improve the applicability and suitability of LOM for resolving crosscutting concerns in real world projects.

Categories and Subject Descriptors D.2.11 [*Software Engineering*]: Software Architectures—Domain-specific architectures

General Terms Language, Design.

Keywords Language oriented modularity (LOM), Aspect oriented programming (AOP), Domain specific aspect language (DSAL), General purpose aspect language (GPAL).

1. Introduction

In principle, programming with DSALs using the LOM methodology can be more effective than programming with General-Purpose Aspect Languages (GPALs), since not only does it promote separation of crosscutting concerns like GPALs do, but it also promotes programming with more declarative and easier-to-use languages. In practice, however, the high implementation cost of DSALs renders LOM cost-ineffective. For LOM to be practical, the cost of implementing composable DSALs must be reduced.

Our key observation is that a more restrictive form of LOM can be applied without needing to modify the weaver (aspect compiler) for each new DSAL. Modifying the weaver to support the weaving semantics of each DSAL has significant implications. First, it requires low level programming and bytecode manipulation, a difficult task for many developers. This increases the cost of implementing a DSAL. Second, it breaks the compatibility of existing development tools with DSAL code. This increases the cost of using a DSAL. We therefore strive for a development process of composable DSALs that would eliminate the need for compiler modifications.

2. Approach

Our approach is to bypass per-DSAL weaver modification by transforming DSALs into a predefined kernel language. The kernel language is based on a GPAL whose weaving semantics is rich enough for a wide range of DSALs. We enhance the GPAL with metadata constructs for specifying how to resolve foreign advising and co-advising conflicts that might occur when multiple DSALs are used simultaneously. The compiler of the kernel language then weaves the code according to the metadata constructs.

With our approach, the DSAL development process becomes similar to that of DSLs, and can be completed using a language workbench. Development tools for the GPAL can work properly with DSAL code. Combined with general editing tools that can be generated by the language workbench for programming in the DSAL and the ability of the compiler of the kernel language to compile DSAL code directly (without the language workbench), programming with DSALs becomes more effective. This in turn improves the cost-effectiveness of LOM.

We apply the approach to modularize scattered and tangled code in the oVirt open source project. We show that our approach leverages DSL development tools in easing the development of DSALs, and thus demonstrates the effectiveness of LOM. Our approach is similar to that of SPECTACKLE, which introduced a specification based approach to DSALs definition. However, we apply it at the level of the code transformation, while SpecTackle applied it to the configuration of a pluggable weaver.

3. Summary

In sum, we address the Achilles heel of LOM, namely the cost of implementing DSALs. By reducing that cost, our approach could increase the adoption of LOM (and AOP) in real world projects. A limitation of our approach is its reliance on a transformation of the DSALs into a GPAL-based kernel language. This means that it is applicable only to DSALs that are in a sense reducible to that GPAL.

References

[1] Arik Hadas and David H. Lorenz. Application-specific language-oriented modularity: A case study of the oVirt project. In *MASS'16*, Málaga, Spain, March 2016.

[2] Arik Hadas and David H. Lorenz. Toward disposable domain-specific aspect languages. In *FOAL'16*, Málaga, Spain, March 2016.

[3] Arik Hadas and David H. Lorenz. Toward practical language oriented modularity. In *LaMOD'16*, Málaga, Spain, March 2016.

* This research was supported in part by the *Israel Science Foundation (ISF)* under grant No. 1440/14.