The Journal of Systems and Software 83 (2010) 1078–1093

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Automating the construction of domain-specific modeling languages for object-oriented frameworks

André L. Santos [a,*], Kai Koskimies [b], Antónia Lopes [a]

[a] Department of Informatics, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisboa, Portugal
[b] Department of Software Systems, Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland

ARTICLE INFO

Article history:
Received 10 February 2009
Received in revised form 31 August 2009
Accepted 27 January 2010
Available online 4 February 2010

Keywords:
Domain-specific modeling
Object-oriented frameworks
Software product-lines
Aspect-oriented programming

ABSTRACT

The extension of frameworks with domain-specific modeling languages (DSML) has proved to be an effective way of improving the productivity in software product-line engineering. However, developing and evolving a DSML is typically a difficult and time-consuming task because it requires to develop and maintain a code generator, which transforms application models into framework-based code. In this paper, we propose a new approach for extending object-oriented frameworks that aims to alleviate this problem. The approach is based on developing an additional aspect-oriented layer that encodes a DSML for building framework-based applications, eliminating the need of implementing a code generator. We further show how a language workbench is capable of automating the construction of DSMLs using the proposed layer.

1. Introduction

Object-oriented frameworks are an important means for realizing *software product-lines* (Bosch, 2000) as they allow partial design and implementation solutions to be defined for families of applications. In this situation, the individual software products of the product-line are developed by *instantiating* the corresponding framework. The activities related to developing a framework are known as *domain engineering*, whereas *application engineering* refers to the development of individual, framework-based applications.

Learning how to correctly use a non-trivial framework is a difficult and time-consuming activity (Moser and Nierstrasz, 1996). The situation is even worse if the framework changes regularly, as happens in the case of frameworks for software product-lines. An effective way of facilitating the task of application engineers is to provide them with a *domain-specific modeling* (DSM) solution for the framework (Kelly and Tolvanen, 2008). The conventional way of realizing a DSM solution involves the development of (i) a domain-specific modeling language (DSML) that captures the conceptual variability of the family of applications that can be built with the framework, and (ii) a code generator for generating full working framework-based applications from the models. We refer to such realizations of DSM as conventional approaches.

Using model-driven engineering terminology, a DSML can be defined by a *meta-model*, whereas *application models* can be defined as instances of the meta-model. Meta-models and required generators can be developed using the so-called *language workbenches* (Fowler, 2008), which are tools targeted for developing domain-specific development environments, such as MetaEdit+ (MetaCase, 2008), Microsoft DSL Tools (Greenfield and Short, 2005), or Eclipse-based technologies (Eclipse Foundation, 2009). Fig. 1 illustrates the conventional approach for having a DSM solution for a framework (Kelly and Tolvanen, 2008). At the problem domain side, the meta-model describes domain concepts, whereas an application model describes instances of those concepts. At the solution side, the object-oriented framework provides an adaptable and partial implementation that is tailored and filled out by application-specific code, which is generated from application models.

DSM approaches claim that it is possible to increase productivity in application engineering activities by up to an order of magnitude (Kelly and Tolvanen, 2008). However, these productivity gains imply a significant additional effort in domain engineering activities, since the meta-model and the code generator have to be developed and maintained as the framework evolves. A DSML is the result of several development iterations, and nevertheless, new increments have to be developed when the domain evolves, implying modifications in the framework, meta-model, and generators. This makes the evolution of the DSM solution challenging.

The difficulty of building and maintaining a DSM solution stems essentially from the complexity of the mapping between the concept instances expressed in the DSML and the code that has to be

* Corresponding author. Tel./fax: +351 912483944.
E-mail addresses: andre.santos@di.fc.ul.pt (A.L. Santos), kai.koskimies@tut.fi (K. Koskimies), mal@di.fc.ul.pt (A. Lopes).

0164-1212/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.01.047

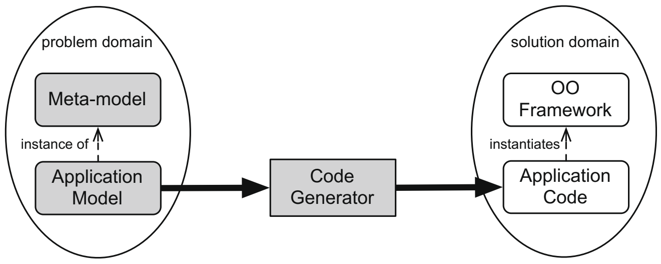

Fig. 1. Conventional realization of DSM solutions.

generated. In principle, the simpler the mapping is, the easier it will be to implement and evolve the code generator. As a matter of fact, DSM approaches are pointed out to be particularly suited to *black-box* frameworks (Roberts and Johnson, 1997), given that in this case the code generation is confined to glue code that composes default components. However, even though the rules for generating such glue code are typically fairly straightforward, they cannot be inferred automatically on the basis of the framework code, because the mechanisms for instantiating the framework-provided concepts are not explicitly represented in the framework implementation. A code generator has thus to be manually developed for implementing this mapping.

In this paper we propose a new approach for developing DSM solutions for object-oriented frameworks based on the extension of the framework with an additional layer, which we refer to as the *DSM layer*. The idea of this layer is to encode the DSML meta-model and all the information that is needed to generate application-specific code from application models. The paper further shows how the DSM layer can be realized using aspect-oriented programming (AOP), capitalizing on our previous work (Santos et al., 2007) that proposes a technique based on AOP for modularizing framework *hot-spots* through (*framework*) *specialization aspects*. In principle, the realization of the approach with AOP is directly applicable to *any* object-oriented framework in its existing form (i.e. there are no special requirements and no need of modification).

The proposed realization of the DSM layer consists of several specialization aspects annotated with additional meta-data for enabling both the meta-model and the mapping between application models and framework-based code to be inferred. We shall show that this can be achieved by means of a generic language workbench, which on the one hand extracts meta-models from DSM layers, while on the other hand is capable of processing instances of those meta-models for generating application code.

The proposed language workbench was implemented in an eclipse-based (Eclipse Foundation, 2009) tool named ALFAMA (Santos, 2008). The tool supports DSM layers written in AspectJ (Eclipse Foundation, 2009) and uses the eclipse modeling framework (EMF) (Eclipse Foundation, 2009) for describing meta-models and application models. As a proof-of-concept, we have implemented the proposed DSM layer for the eclipse rich client platform (RCP) framework (McAffer and Lemieux, 2005), a framework for developing stand-alone applications based on eclipse's dynamic plug-in model and UI facilities, such as menus, actions bars, tree-views, etc. We have tested ALFAMA by developing sample applications that make use of the eclipse RCP features that were included in the developed DSM solution.

Comparing to the state-of-the-practice, the approach proposed in this paper embodies a major strategic difference, given that we propose frameworks to have a "built-in" DSML (syntax and semantics) encoded by the DSM layer. Domain engineers are able to effectively extend a framework's implementation with the encoding of a DSML, which can be directly used to build framework-based applications, without the need of having a code generator. In this way, domain engineers are relieved of the maintenance problems that are typically associated to the development of code generators.

The paper proceeds as follows. Section 2 presents an overview of our approach. Section 3 explains how conventional framework hot-spots can be represented in a modular way in terms of specialization aspects. Section 4 addresses the development of the DSM layer using specialization aspects. Section 5 explains how DSMLs can be automatically derived from DSM layers. Section 6 presents the ALFAMA tool. Section 7 compares the proposed approach with conventional approaches for DSM. Section 8 describes the case study on eclipse RCP. Section 9 discusses related work, and Section 10 concludes the paper.

2. Approach overview

This section presents an overview of our approach (see Fig. 2). The approach relies on a language workbench that automates the DSML construction and usage at the expense of some new development activities. The following summarizes the role of the different elements involved in the process of developing the DSM solutions using our approach, making a comparison with the conventional approach (depicted in Fig. 1):

- *Domain engineers*. Domain engineers have to develop a DSM layer in addition to the framework, while they are relieved of both implementing a code generator and defining the DSML concepts separately (i.e. externally to the framework implementation).
- *Language workbench*. The language workbench extracts the meta-model that defines the DSML from the DSM layer. Additionally, it is able to transform application models into code based on the DSM layer. The language workbench is generic, in the sense that it can be used for multiple frameworks, as long as the DSM layers are developed in the supported programming language (in this work, AspectJ for Java frameworks).
- *Application engineers*. Application models are developed by application engineers as if a conventional approach was being used. Although the generated code is based on the DSM layer instead of directly on the framework, application models are developed in the same way.

The DSM layer includes a formal representation of the meta-model embedded in its modules, using modeling constructs that are equivalent in terms of expressiveness to those that can be found in existing meta-modeling technologies (MetaCase, 2008; Greenfield and Short, 2005; Eclipse Foundation, 2009). When implementing the ALFAMA prototype as a proof-of-concept of the proposed language workbench, our option was to consider the modeling constructs that are available in EMF (Eclipse Foundation, 2009). EMF is a Java implementation of the meta-object facility (MOF) (OMG, 2002), a standard for defining modeling

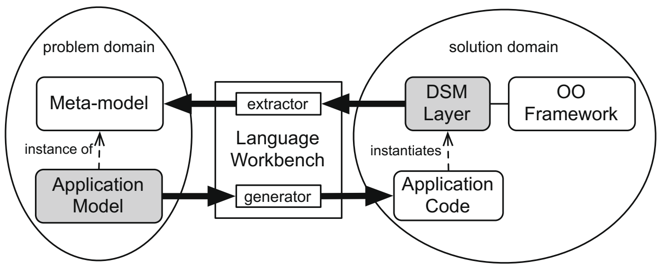

Fig. 2. Proposed approach for realizing DSM solution.