Reverse Engineering of Model Transformations for Reusability

Jesús Sánchez Cuadrado, Esther Guerra, and Juan de Lara

Modelling and Software Engineering Research Group
Universidad Autónoma de Madrid, Spain
http://www.miso.es

Abstract. Reuse techniques are key for the industrial adoption of Model-Driven Engineering (MDE). However, while reusability has been successfully applied to programming languages, its use is scarce in MDE and, in particular, in model transformations.

In previous works, we developed an approach that enables the reuse of model transformations for different meta-models. This is achieved by defining reusable components that encapsulate a generic transformation template and expose an interface called concept declaring the structural requirements that any meta-model using the component should fulfil. Binding the concept to one of such meta-models induces an adaptation of the template, which becomes applicable to the meta-model. To facilitate reuse, concepts need to be concise, reflecting only the minimal set of requirements demanded by the transformation.

In this paper, we automate the reverse engineering of existing transformations into reusable transformation components. To make a transformation reusable, we use the information obtained from its static analysis to derive a concept that is minimal with respect to the transformation and maximizes its reuse opportunities, and then evolve the transformation accordingly. The paper describes a prototype implementation and an evaluation using transformations from the ATL zoo.

Keywords: Model transformation, Reusability, Reverse engineering, Re-engineering.

1 Introduction

Reusability is a key enabler for the industrial adoption of Model-Driven Engineering (MDE). Some techniques have been proposed to reuse complete transformations, such as superimposition [19], phases [14] and genericity [13], but their use is still an exception. As noted by [1], one reason for this situation is the lack of repositories for selecting and effectively reusing transformations. Even the ATL Transformation Zoo [2], which is the closest relative to a transformation repository, consists of a collection of transformations not designed for reuse. This contrasts with the rich ecosystems of libraries in e.g., object-oriented languages like Java or C#, which successfully promote development with reuse.
In previous works [13], we proposed a technique for transformation reuse based on generic programming. In our approach, reusable transformation components encapsulate a transformation template developed against so-called concepts, which resemble meta-models but their elements are variables. Binding these variables to concrete meta-model elements induces a rewriting of the template to make it compatible with the meta-model. Thus, we obtain reusability because the transformation component can be used with any meta-model that can be bound to its concepts. However, this technique implies developing transformations with reusability up-front, by designing suitable concepts for the input and output domains and then writing the transformation template accordingly. Thus, it is not possible to profit from existing transformations beyond their use as a reference to manually implement a generic, reusable transformation. While concepts need to be concise to facilitate reuse and include only the elements accessed by a template, transformations are developed for concrete meta-models (e.g. UML) which reflect the complexity of a domain and may include accidental complexity from the transformation point of view. Hence, making an existing transformation reusable requires both a simplification of the meta-model into a truly reusable concept, and an according reorganization of the transformation.

In this work, we propose a semi-automatic process to reverse engineer existing transformations into generic, reusable transformations. It has been implemented for ATL as this is one of the most widely used transformation languages. Our aim is to foster reuse by facilitating the transition from existing, non-reusable transformations into reusable components that can be offered as transformation libraries in a repository. The process starts by extracting the effective meta-model of a transformation, which implies its static analysis to derive typing information. Then, the effective meta-model is evolved towards a concise concept through a series of refactorings, and the transformation is co-evolved accordingly if needed. The approach is supported by a prototype tool, and has been evaluated using transformations of the ATL zoo.

Organization. Section 2 presents our previous work on reusable transformations. Then, Section 3 overviews our proposal to the reverse engineering of existing transformations into reusable components, which is detailed in the following two sections: static analysis of ATL transformations (Section 4), and extraction and customization of concepts (Section 5). We evaluate our approach in Section 6, review related work in Section 7, and draw conclusions in Section 8.

2 Reusable Transformations

In order to build a reusable transformation, in previous work [13] we proposed the notion of transformation components with a well-defined interface called concept. Fig. 1 shows a generic transformation component to calculate metrics for object-oriented languages, as well as its instantiation for a specific meta-model. The component (label 1) includes a transformation template from a hand-made concept characterising object-oriented languages to a metrics meta-model. We only show an excerpt of the template, which calculates the Depth of Inheritance