RubyTL: A Practical, Extensible Transformation Language

Jesús Sánchez Cuadrado¹, Jesús García Molina², and Marcos Menarguez Tortosa³

¹ University of Murcia, Spain
jesusc@um.es
² University of Murcia, Spain
jmolina@um.es
http://dis.um.es/~jmolina
³ University of Murcia, Spain
marcos@um.es

Abstract. Model transformation is a key technology of model driven development approaches. A lot of research therefore is being carried out to understand the nature of model transformations and find out desirable characteristics of transformation languages. In recent years, several transformation languages have been proposed.

We present the RubyTL transformation language which has been designed as an extensible language—a set of core features along with an extension mechanism. RubyTL provides a framework for experimenting with features of hybrid transformation languages. In addition, RubyTL has been created as a domain specific language embedded in the Ruby programming language. In this paper we show the core features of the language through a simple example and explain how the language can be extended to provide more features.

1 Introduction

The model-driven development (MDD) promotes an intensive use of models in the software life cycle. Software models are used to guide the construction of the application, and an automatic generation of source code from models is possible. At the end of 2000, OMG launched its initiative on the Model Driven Architecture™ (MDA) [1], an MDD approach to address the integration challenges and the continuous changes in technology. Since then other approaches have been proposed [2] [3] [4], and MDD has become the new software paradigm that promises to improve software productivity and quality.

Model-to-model transformations are a key technology of the MDA approach. Most MDA research has been focused on understanding the nature of transformations and discovering desirable characteristics of model transformation languages and tools. In recent years, several transformation languages have been defined [5] [6] [7], among them the QVT [8] standard proposed by the OMG. Today the success of QVT is not clear, and an alternative of a set of languages providing different styles makes sense [9].

© Springer-Verlag Berlin Heidelberg 2006
In this paper we present RubyTL, a hybrid transformation language which has been designed with three main requirements in mind: i) rapid implementation, ii) it should allow us to experiment easily with different sets of features, iii) it should provide enough functionality for writing complex transformation definitions. Three design decisions have allowed us to satisfy these requirements: the technique of embedding a domain specific language (DSL) in a programming language such as Ruby facilitates the implementation; a plugin mechanism provides a way of adding extensions, so that the language may be configured to experiment with different sets of features; finally, Ruby constructs could be used to write some kinds of complex transformations, in which a declarative style is not the most suitable. In short, RubyTL is an extensible language which provides a set of core features and an extension mechanism to add new features.

The paper is organized as follows. Section 2 describes the basic features of RubyTL transformation language, while Section 3 shows the extension mechanism. In Section 4 the transformation process is discussed. Section 5 compares RubyTL with other proposed languages. Finally, in the last section we present our conclusions and outline future work.

## 2 Language Description

In this section we explain the RubyTL core features, and use a transformation definition example between two simple models to illustrate the syntax and semantics of the language. These features are the basic ones for a usable transformation language, but they can be extended, as explained in Section 3.

Ruby [10] is an object-oriented programming language which is gaining constantly acceptance, especially over the last year because of the success of Ruby on Rails, a web application framework. Ruby is dynamically typed and provides an expressive power similar to Smalltalk through constructs such as code blocks and metaclasses. Because of these characteristics, Ruby is very suitable to define internal DSLs [3].

Thus, RubyTL is a model transformation language defined as a Ruby internal DSL. RubyTL is a hybrid language since it provides both declarative and imperative constructs to write transformation definitions. Like ATL [6][9], a binding construct is used to express rules in a declarative way.

The RubyTL abstract syntax, expressed as a metamodel, is shown in Figure 1. As can be seen, a transformation definition is a set of transformation rules packaged in a transformation module, and each rule has a name and four parts:

- A *from* part, where the source element metaclass is specified.
- A *to* part, where the target element metaclass (or metaclasses) is specified.
- A *filter* part, where a condition over the source element is specified, such that the rule will only be triggered if the condition is satisfied; this part is optional and if a rule has no filter it will always be triggered.