Science of Computer Programming 92 (2014) 86–104

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

# Consistency of model transformation contracts

Christiano Braga [a,b,*], Cássio Santos [a,b], Viviane Torres da Silva [a]

[a] Instituto de Computação, Universidade Federal Fluminense, Brazil
[b] ADDLabs, Universidade Federal Fluminense, Brazil

ARTICLE INFO

*Article history:*
Received 7 April 2012
Received in revised form 8 August 2013
Accepted 19 August 2013
Available online 5 September 2013

*Keywords:*
Mathematical aspects of software engineering

ABSTRACT

Model-driven development is a generative software development process with increasing relevance both in industry and academia. Model transformations are the generative components in a model-driven development process. As such, their analysis is an important task. We have been developing a technique to specify, validate and implement model transformations. Our technique is based on the concept of *transformation contracts*, a specification that relates two modeling languages and declares properties that must be fulfilled in such a relation. Since a transformation contract is a model, the verification and validation of a transformation contract use the same techniques that are used to verify and validate any given model. This paper describes our technique, discusses *consistency* of model transformations and reports on its application to a non-trivial model transformation from access control models to Java security.



## 1. Introduction

Model-driven development (MDD, e.g. [1]) is a software engineering discipline that considers models as *live artifacts* in the development process. By live artifacts we mean that models are not used for documentation purposes only but actually as input to software tools that may operate on them and produce other artifacts. Such artifacts may be compilable source-code or even other models, in the same or different abstraction levels than the source model. MDD aims at allowing for a *generative* software development process in which applications are produced from formal models possibly described at the application domain level.

Model transformations are important artifacts in a model-driven development process since they are the generative components of the process. As such, their specification, verification and validation are imperative tasks in an MDD process. A transformation contract (e.g. [2–5]) is a specification of a model transformation. Essentially, a transformation contract is comprised by *relations*, between the model elements of the modeling languages it relates, and *properties* that such relations must fulfill. Therefore, a model transformation specification may be understood as a model that relates metamodels. (More specifically, it is the disjoint union of metamodels, as we shall see in Section 4.) In this paper, inspired by [6], we call the model representing a model transformation a *transformation model*. A particular application of a model transformation is represented as an object model instance of the transformation model. A transformation contract is thus a transformation model and a set of properties over it. Therefore, one may use the *same modeling language* used to specify the modeling languages being related to also describe the model transformation and the *same model reasoning* techniques may be applied to *reason about model transformations* as well.

In [2–5,7,8], the present authors and others specify metamodel properties as invariants in the Object Constraint Language representing *typing rules* of the different metamodels involved in a model transformation. In this paper, we generalize

* Corresponding author at: Instituto de Computação, Universidade Federal Fluminense, Brazil.
*E-mail addresses:* cbraga@ic.uff.br (C. Braga), cfernando@ic.uff.br (C. Santos), viviane.silva@ic.uff.br (V.T. da Silva).

0167-6423/$ – see front matter 
http://dx.doi.org/10.1016/j.scico.2013.08.013

previous work and allow for the automatic verification of *model transformation consistency* understood as satisfiability of an associated theory in Description Logics [9].[1] We also discuss the implementation of transformation contracts as the application of a *design pattern* [5] extended with consistency checking support. As a proof of concept, we discuss the application of our proposed technique with a non-trivial model transformation from access control models to Java security.

This paper contributes with: (i) a *logical* formalization of transformation contracts, a rigorous approach to the specification, verification and implementation of model transformations, (ii) the inclusion of model consistency verification into the transformation contracts approach, and (iii) tool support to the proposed approach, *vis-à-vis*, (a) the *Consistency Checker*,[2] a tool for checking model consistency according to the technique discussed in Section 4, and (b) the *SecureUMLtoAAC+JAAS* tool,[3] an implementation of the model transformation from SecureUML to Java security, that performs OCL validation and consistency checking for stereotyped SecureUML models.

This paper is a revision and an extension of [10]. All sections have been revised, made more precise and extended. Section 4, in particular, was thoroughly extended with new results from the formalization of the notion of extended TBoxes of models while reasoning about model transformation consistency with Description Logics.

*Plan of the paper*. Section 2 discusses related work. Section 3 describes our proposed model transformation process, making precise important concepts in MDD that are otherwise loosely applied. Section 4 formalizes transformation contracts as theories in Description Logics and describes how consistency verification may be added to our model transformation process. Section 5 describes our case study. It reports on the rigorous implementation of model transformations according to our proposed model transformation process and exemplifies its application. We conclude this paper in Section 6 with our final remarks.

## 2. Related work

In this paper we discuss three aspects of the development of model transformations, namely its specification, verification and implementation. Therefore, we organize this section following these three perspectives.

At the specification level, we adopt a *relational* approach towards the specification of a model transformation. It is similar in essence to [11,6] but different from [2,3]. In [2] the authors specify transformation contracts as OCL invariants from source and target model elements. We formalize our understanding of transformation contracts in Section 3 as opposed to the informal discussion in [11,6,3]. (The authors in [2,3] also discuss the use of pre- and post-conditions but such predicates may also be represented as invariants.) The specification of a relation between the model elements of the metamodels related by a model transformation is essential to generalize from OCL invariants and understand that different *kinds* of properties may be specified *over such relation*. It is important to make *explicit* the relationship among the metamodels.

In [6] the idea of transformation model to specify model transformation is discussed. The authors describe the benefits of omitting details of the transformation process and concentrating in depicting the transformation as a model, in which models are instances of metamodels and transformations can be described in conformity to a metamodel. From this point it is possible to describe a transformation model in a formal way, thus it can be validated and verified. We share the idea of transformation models but in this work we make precise what we mean by transformation model (as the result of a disjoint union of two metamodels) and how it may be used to reason on model transformations, as opposed to [6].

OMG's Query View Transformations (QVT) [12] and Graph Grammars (GG) (e.g. [13–15]) are other possible specifications for a model transformation that require specific theory and machinery to reason and implement model transformations. (In the case of GG, focus is on the verification of typical properties of term rewriting systems such as confluence and termination.) The transformation contracts approach proposed in this paper is not biased by any particular specification language and may be used with them as well. Note, however, that QVT is bound to OCL as the specification language for the properties of metamodels and there are properties, such as consistency, subject of this paper, best specified in other semantic frameworks. GG may not have QVT's restriction on the specification language for metamodel properties but a key aspect of our approach is that we apply to model transformation specification design the same specification languages and techniques one would to design a modeling language. No additional framework, such as QVT or GG, is necessary.

At the verification level, different kinds of properties may be reasoned upon besides OCL invariants. One such property is model consistency understood as satisfiability of a propositional formula representing a constrained model. In this paper, we check for the consistency of Description Logic theories associated with a given model. Note that, since we understand model transformations as models, their consistency may be checked as one would check the consistency of any given model, such as in [16].

In [17] the authors use an SMT-solver to check for the satisfiability of model transformations. The mapping is essentially built upon a first-order logic semantics for OCL in [18]. However, they are focused on a particular model transformation language. We believe that to be able to use the same modeling language to describe both the metamodels and model transformation is an important issue.

[1] In this paper, we focus on a particular Description Logic that allows us to write axioms encompassing set union, set intersection, set complement, and binary relations with inverse and cardinality constraints.

[2] http://lse.ic.uff.br/node/11.

[3] http://lse.ic.uff.br/node/14.