Journal of Visual Languages and Computing 22 (2011) 233–250

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

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

Using rule overriding to improve reusability and understandability of Dynamic Meta Modeling specifications[☆]

Christian Soltenborn [*], Gregor Engels

University of Paderborn, Warburger Straße 100, 33098 Paderborn, Germany

ARTICLE INFO

Article history:
Received 14 May 2010
Received in revised form
25 October 2010
Accepted 8 December 2010
Available online 12 January 2011

Keywords:
Semantics
Metamodel
Dynamic Meta Modeling
Graph transformation
Inheritance

ABSTRACT

Dynamic Meta Modeling (DMM) is a visual semantics specification technique targeted at languages based on a metamodel. A DMM specification consists of a *runtime metamodel* and *operational rules* which describe how instances of the runtime metamodel change over time. A known deficiency of the DMM approach is that it does not support the refinement of a DMM specification, e.g., in the case of defining the semantics for a refined and extended domain-specific language (DSL). Up to now, DMM specifications could only be reused by adding or removing DMM rules.

In this paper, we enhance DMM such that DMM rules can *override* other DMM rules, similar to a method being overridden in a subclass, and we show how rule overriding can be realized with the graph transformation tool GROOVE. We argue that rule overriding does not only have positive impact on reusability, but also improves the intuitive understandability of DMM semantics specifications.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

In today's software engineering world, typically a lot of people with different backgrounds are involved in a software project. This is due to the increased complexity of software, but also because software systems are used in basically every area; as a consequence, software engineers have to work together with domain experts from different fields.

One way to support communication between those differently skilled groups of people is to use visual modeling languages. This approach works best if the language that is used supports concepts of the actual domain; so-called *Domain-Specific Languages* (DSLs) might even enable the domain experts to do large parts of the modeling themselves (in contrast to a general-purpose language like the *Unified Modeling Language* (UML) [1]).

One requirement for effectively using DSLs is a precise definition of the language's semantics. This is often the case for the *static semantics* of a language (the UML is no exception here): Valid sentences of a language are e.g., described by means of *metamodels*, i.e., class diagrams describing the language's concepts as well as their relation to each other. Additional, context-sensitive constraints—which cannot be expressed with class diagram constructs—are added, e.g., using a language like the *Object Constraint Language* (OCL) [2].

For the definition of the *dynamic semantics* of a language, the situation is not as good. Often, natural language is used to describe how models of a certain language behave. For instance, the UML specification states that the semantics of Activities is based on token flow, but this information is only contained in the text accompanying the definition of the static semantics.

Such an informal description of the language's behavior almost always leaves room for different interpretations and is therefore in conflict with the requirement

[☆] This paper has been recommended for acceptance by Mark Minas.
[*] Corresponding author. Tel.:+495251603959; fax:+495251603431.
E-mail addresses: christian@uni-paderborn.de (C. Soltenborn), engels@uni-paderborn.de (G. Engels).

1045-926X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2010.12.005

that the language's meaning needs to be defined precisely. Additionally, other requirements which are often put on DSLs are seriously affected: First, to effectively work with a DSL, a sophisticated *tool support* is needed, and second, the quality of complex models cannot be checked manually; therefore, the language should be *analyzable*.

These requirements can only be fulfilled by a language whose syntax and semantics are specified formally. Unfortunately, formal specifications often lead to another problem: They are difficult to understand for language users who are not familiar with the underlying formalism. This has two severe drawbacks: First, the language engineer's job of creating the semantics specification is more difficult, and second, end users of the language cannot refer to the semantics specification as a reference when, e.g., discussing the semantics of a particular element.

Consequently, a semantics specification technique is needed which is not only formal, but is also easily understandable at least for the target language users (users who are at least familiar with the language's metamodel). This is where *Dynamic Meta Modeling* (DMM) [3,4] comes into play.

DMM specifications are easily understandable for a number of reasons: Firstly, as we will see later, a DMM rule is an (annotated) object diagram instantiating the runtime metamodel; this visual and familiar appearance has proved to be easy to comprehend [5]. Secondly, DMM supports a number of object-oriented concepts, which are expected to be well-known by the target language users.

DSLs are often developed incrementally, i.e., an existing DSL is modified to suit the needs of another but related domain. Therefore, if a DSL has been equipped with a formal specification of syntax and semantics, it is desirable to be able to reuse that specification for the modified DSL.

In its current state, existing DMM specifications can be reused as a base for similar languages, but only with strong restrictions: DMM rules can be added or removed from a DMM specification, but rules cannot *refine* other rules (similar to a class refining methods of its superclasses).

It turned out that this severely hampers reusability of DMM specifications. As a consequence, we decided to introduce a notion of *rule overriding* into the DMM language. Within this paper, we introduce this concept of rule overriding.

In the next section, we give a brief overview on related work and point out the differences between the existing approaches and ours. Section 3 provides an introduction to DMM in its current state, illustrating the different parts of a DMM specification. Since we expect UML activities to be familiar to the readers of our paper, we use a simplified version of that language as a running example. Section 4 then introduces and discusses our notion of rule overriding. The last section concludes and gives an outlook on future work.

This paper is an enhanced version of [6]. In particular, a detailed description of the implementation of rule overriding using the GROOVE toolset [7] has been added to Section 4, more elaborated examples have been provided, and the related work has been enhanced.

2. Related work

This section will discuss research which is related to ours. There are several approaches to define the semantics of visual modeling languages by means of (typed) graph transformations (see e.g., [8,9]). We first do a brief comparison to generic frameworks suited for semantics specification. Next, we will discuss work which provides some support for reusability, such as modularization, prioritization, or support for inheritance. Finally, we will compare DMM to common rule-based model transformation approaches.

One approach to specifying behavioral semantics of visual languages has been suggested by Chen et al. [10]: They define the semantics of a language by *anchoring* it to another language (which they call *semantic unit*) whose semantics is well-defined. The advantage is that the language engineer does not have to come up with an own semantics for his language – instead, he only has to map his language constructs to the ones of the semantic unit. However, this only works if the two languages are closely related to each other; consequently, the authors claim that an appropriate set of semantic units covering the needs of the most common behavioral languages is yet to be identified. In contrast, DMM leaves the burden of specifying the operational semantics to the language engineer, but provides more flexibility, since there are no restrictions on the semantic domain (i.e., the runtime metamodel).

Bottoni et al. [11,12] define *action patterns* and their generalization *generative patterns* for the sake of specifying "semantic building blocks", for instance in the context of token flow. A target language's metamodel is then mapped to a metamodel of the semantic domain, and the defined patterns are instantiated in the context of elements of the semantic domain; the result are rules describing the elements' semantics (i.e., rules which make elements of the target language flow like tokens). Again, the approach only works if appropriate patterns are available which reflect the target language's intended semantics.

We now turn to approaches which directly or indirectly support reusability. The first area of interest we discuss is notions of inheritance. In de Lara et al. [13], show how to integrate attributed graph transformations with node type inheritance, therefore allowing to formulate *abstract* graph transformation rules (i.e., rules which contain abstract nodes). The resulting specifications tend to be more compact, since a rule containing abstract nodes might replace several rules which would otherwise have to be defined for each of the concrete subtypes. The resulting formalism does not provide support for the refinement of rules (and is therefore comparable with the expressiveness of the current state of DMM).

Ferreira et al. [14,15] develop a notion of typed graph transformations which supports several object-oriented features, including inheritance and polymorphism. They focus on delivering a framework which is as close to object-oriented systems described by e.g., Java code as possible, whereas the models targeted by DMM are expected to have a less complex semantics, since they

Download English Version:

https://daneshyari.com/en/article/523572

Download Persian Version:

https://daneshyari.com/article/523572

Daneshyari.com