Code Generation by Object Observation - an Evaluation*

Sebastian Geiger[1], Sebastian Kerekes[2], Michael Kraxner[3], and Martin Lackner[4]

[1] Favoritenstrasse 9-11, 1040 Wien
sbastig@gmx.net
MatrNr.:
[2] Favoritenstrasse 9-11, 1040 Wien
contact@sebastiankerekes.com
MatrNr.:
[3] Favoritenstrasse 9-11, 1040 Wien
michael.kraxner@gmail.com
MatrNr.: 9925916
[4] Favoritenstrasse 9-11, 1040 Wien
lackner.martin@gmail.com
MatrNr.: 0927551

Abstract. ... This paper evaluates possibilities and limitations of code generation by object observation. ...

* This work has been created in the context of the course "Advanced Model Engineering" (188952) in SS13.

Table of Contents

1 Introduction

In the last decade model driven engineering has devop ...

2 Semantics

Any (model) language is defined by its syntax, in this context a metamodel, and its semantics, the meaning of each element in the metamodel. We can distinguish between the static and the dynamic semantics. The static semantics defines structural properties of model elements in a metamodel (for further reading see [1] [3]), and the dynamic semantics regards to the execution of such models and its behavior [3]. For our work we are interested in the dynamic semantics.

2.1 Dynamic Semantics

The dynamic or execution semantics defines the behavior under execution of models. This can be done in many ways, e.g., the semantics of UML is defined in OCL and natural language. A more sophisticated way which is seen very often in practice is either to use code generators to generate compilable source code, or to provide an interpreter which can process the models directly. But this approach is only feasible if the semantics is defined formally. Otherwise it would be too error prone, time-consuming and very costly if this would be done by manually studying and exploring the models [3].

We can distinguish between four different "semantic description styles" to define the semantics of a model formally [3] [1]:

Denotational
: This semantic provides a mapping between mathematical functions and model specific language constructs. E.g., it maps the keyword "-" to the subtraction function on natural numbers.

Axiomatic
: This semantic provides a mapping between logical rules and model specific language constructs. E.g., stating that two arguments of a function "-" produces an output with the same type of the two arguments.

Translational
: This semantic provides a mapping between the elements of a original model and model elements of a language where the semantics is already formally defined. The mapping is done by model transformation and can be used if the semantics of the source and the target models are closely correlated. Due to the fact, that the model transformation is done on the syntax level it is not guaranteed that the semantic of the final model behaves as desired. You also have to have an in-depth knowledge of both model languages and use multiple frameworks to achive good results.

Operational
: This semantics describes how model constructs are executed on an abstract machine. It describes the effect of the computation and how the computation is produced.

Another approach to define the semantics is to "weave behavior" into the abstract syntax by a meta-language. On this approach the behavior of an operation is specified inside the body of the operation on the meta level. E.g., Kermata uses this approach by extending its abstract meta-layer with an imperative action language [2].

The last approach we will mention here is to define semantics with "Rewrite Rules". A system consits of rewrite rules and each rewrite rule has a left- and a right-hand side. When executing a model the system takes an model element, looks if such a model element occurs on the left-hand side of a rule and replaces it with the right-hand side if such a configuration is found [2].

3 Related work

recent developments ... Executable UML,

3.1 fUML

3.2 xMOF

3.3 xtend

XXX really ???

4 Code Generation with xtend

5 Code Generation by Object Observation

6 Evaluation

References

1. A. Cuccuru, C.. Mraidha, F. Terrier, S. Gerard. Enhancing UML Extensions with Operational Semantics. *Lecture Notes in Computer Science Volume 4735* (2007), 271–285.
2. B.R. Bryant, J. Gray, M. Mernik, P.J. Clarke, R.B. France, G. Karsai. Challenges and directions in formalizing the semantics of modeling languages. *Computer Science and Information Systems 2011 Volume 8, Issue 2* (2011), 225–253.
3. S. Andova, M.G.J. van den Brand, L.J.P. Engelen, T. Verhoeff. MDE Basics with a DSL Focus. *Lecture Notes in Computer Science Volume 7320* (2012), 21–57.