Transformational Techniques in Model-Driven (Language) Engineering

Guido Wachsmuth
first supervisor: Prof. Schlingloff
second supervisor: Dr. Holz
in METRIK since: 2006-06-15

Problem statement

Transformational techniques are a fundamental building block in Model-Driven Engineering. Model transformations are employed to automate various software engineering tasks including synthesis of more abstract into more specific models, code generation, migration, optimisation, refactoring, and reverse engineering. Model-Driven Language Engineering applies techniques from Model-Driven Engineering to the definition and implementation of modelling languages. Here, model transformations are used in two different ways: First, they are instrumented to derive various software tools from language models. For example, we can generate an editor from a model describing the syntax of a modelling language. Second, they provide means to model certain aspects of modelling languages. For example, we can model the semantics of a modelling language as a transformation which translates this language into a target language.

In my thesis, I focus on several open problems in the context of transformational techniques in Model-Driven (Language) Engineering: (1) Text-to-model transformation languages are widely used to specify the textual concrete syntax of modelling languages. But these languages lack means to specify non-trivial analyses like type inference or name resolution. Instead, these analyses need to be defered to a postprocessing transformation which is commonly implemented in a programming language like Java. (2) Template languages provide popular means for code generation. But they lack formal semantics as well as guarantees for the syntactically correctness of generated code. (3) Code generation is the typical approach to model execution. But this is not the proper level for specifying and prototyping language semantics because semantics are intermingled with details of the target language and the target platform. This inhibits understanding, troubleshooting, and adaptation of semantics — particularly for domain experts. (4) Modelling languages evolve. This implies a risk of language erosion: The language, its tools, and existing models do not longer comply.

Approach

In software engineering, there is a long tradition of grammar-based language descriptions and transformational techniques. As a consequence, an exhaustive body of knowledge exists. This includes approaches to solve problems in the grammar world which are quite similar to the problems mentioned above. The aim of my thesis is to adopt these approaches and apply them to problems in the modelling world:

(1) I adopt the concept of attribute grammars to specify text-to-model transformations. From such a specification, it is possible to generate a parser and a decorator. Thereby, the decorator is implemented in QVT Relations, a standard model transformation language. The transformation is then performed in three steps: First, the parser turns the text into a syntax tree. Second, the syntax tree is converted into a syntax tree model. This conversion is implemented generically. Third, the decorator decorates the tree model with elements from the target model.

(2) I provide formal semantics for the core concepts of template languages. Therefor, I rely on Natural Semantics for describing both the static and the dynamic semantics of these concepts. Furthermore, I deal with template languages for code generation in a particular target language. I provide construction steps for the syntax and semantics of such languages. The approach is generic and can be applied to any target language.

(3) I investigate how operational semantics of modelling languages can be described by standard modelling means. The runtime state of a model is threated as a model itself. The set of valid runtime states is specified by a metamodel. Valid transitions between runtime states are specified by a model-to-model transformation. I suggest QVT Relations to express such transformations. Relying on the model-view-controller pattern, the approach can be instrumented for the prototyping of visual interpreters and debuggers.

(4) To assist the evolution of modelling languages, I propose an operator suite for the stepwise adaptation of metamodels. Operators from this suite are implemented as metamodel-to-metamodel transformations. For most operators, model migration can be achieved in a generic way. The approach is evaluated in a case study on the evolution of modelling languages provided by the Eclipse Graphical Modeling Framework.

Abstract

My thesis addresses open problems in Model-Driven Language Engineering by adopting existing solutions in the grammar world to the modelling world.

Publications

1) Markus Herrmannsdörfer, Daniel Ratiu, Guido Wachsmuth: *Language Evolution in Practice: The History of GMF*. SLE 2009.
2) Daniel Sadilek, Guido Wachsmuth: *Using Grammarware Languages to Define Operational Semantics of Modelled Languages*. TOOLS Europe 2009.
3) Guido Wachsmuth: *A Formal Way from Text to Code Templates*. FASE 2009: LNCS 5503, 109-123.
4) Wolfgang Lohmann, Günter Riedewald, Guido Wachsmuth: *Aspect-oriented Prolog in a Language Processing Context*. Software IET 2(3), 2008.
5) Daniel Sadilek, Guido Wachsmuth: *Prototyping Visual Interpreters and Debuggers for Domain-Specific Modelling Languages*. ECMDA-FA 2008: LNCS 5095, 63-78.
6) Guido Wachsmuth: *Modelling the Operational Semantics of Domain-Specific Modelling Languages*. GTTSE 2007: LNCS 5235, 506-520.
7) Guido Wachsmuth: *Metamodel Adaptation and Model Co-adaptation*. ECOOP 2007: LNCS 4609, 600-624.
8) Guido Wachsmuth: *Adaptation Browser for MOF*. WRT 2007.
9) Daniel Sadilek, Falko Theisselmann, Guido Wachsmuth: *Challenges for Model-Driven Development of Self-Organising Disaster Management Information Systems*. International Research Training Groups Workshop 2006.
10) Ralf Lämmel, Guido Wachsmuth: *Transformation of SDF syntax definitions in the ASF+SDF Meta-Environment*. LDTA 2001, ENTCS 44(2).