Abstract

Model Driven Engineering (MDE) promotes the use of models as primary artefacts of a software development process, as an attempt to handle complexity through abstraction, e.g. to cope with the evolution of execution platforms. MDE follows a stepwise approach, by prescribing to develop abstract models further improved to integrate little by little details relative to the final deployment platforms. Thus, the application of an MDE process results in various models residing at various levels of abstraction.

Each one of these models is expressed in a modeling language, in which one may find appropriate concepts for the abstraction level considered. Many advocate to use the right (modeling) language for the right purpose. This means that it is sometimes better approach to use small languages specific to the considered domain and abstraction level, than to use general purpose languages (e.g. UML) when they do not perfectly fit the (modeling) needs. As a matter of fact, an MDE development process, which involves many different domains and abstraction levels, should also involve a large variety of modeling languages. Project managers who want to apply an MDE process need to deal with this language proliferation to such an extent that, in the long run, one may infer that language engineers can become major actors of software development teams.

We believe that comprehensive modeling language management facilities may considerably alleviate that MDE drawback. Such facilities may include modeling language definition, extension, adaptation, or composition. To define a (modeling) language, one need to define its abstract syntax, its semantics, and one or more concrete syntaxes. This thesis focuses on concrete syntax definition for modeling languages, when the abstract syntax is given in the form of a metamodel. We will provide solutions both for textual and graphical concrete syntaxes.

Some of our experiences in building textual languages (as MTL, a model transformation language), and graphical languages (as Netsilon, a web-application modeler) has shown that a lot of work was spent in implementing interface using traditional techniques, be it a text processor generated from a compiler compiler specification, or a modeler making use of modern 2D graphical libraries. Indeed, abstract and concrete syntax were implemented in a disconnected way, and it was then necessary to assemble them, which became rapidly clumsy while abstract syntax evolved.

We built our solution to concrete syntax definition as companions of the abstract syntax. The definition of concrete syntax we propose here made it possible to build automatic tools able to analyze or synthesize models from/to text, and to create graphical modelers. We will present a metamodel for textual concrete syntax definition to construct constructive reversible grammars. We will also propose a technique for graphical concrete syntax definition following a two-step process: specification and realization. Specification is a restrictive approach in which a metamodel defines a graphical concrete syntax. Both relations with

abstract syntax and spatial relationships are expressed by means of constraints. The realization step proposes a way to provide the concrete syntax tree a meaning, by attributing it a graphical appearance, and by expressing possible user interactions.

The structure of the document is the following. After introducing in deeper details the problem and the general structure of the solution we propose, we will take a tour of MDE, text and graph grammars. Then, we will present Netsilon as an example of an MDE tool to MDE development, which required both the definition of a graphical and a textual modeling language. The two following sections will present the solutions we propose for textual and graphical concrete syntax definition, respectively. Final remarks and possible improvements, especially regarding reusability in general of MDE meta-artifacts (like metamodels or model transformations), and of concrete syntax in particular, will conclude the document.

Keywords

Model Driven Engineering, Metamodeling, Language Engineering, Concrete Syntax, Textual Concrete Syntax, Graphical Concrete Syntax, Scalable Vector Graphics.