Textual languages with Xtext

Fedor Biryukov

University of Antwerp

Abstract

Keywords: Modeling, Xtext, DSL, AToMPM, metaDepth

1. Introduction

In his lections [1], Vangheluwe pays a lot of attention to visual concrete syntax and it is almost asserted that domain-specific modeling is done in either visual or hybrid manner. While I am convinced that visual modelling has a lot of advantages and far more expressive power (shape, color etc.) [2], I am eager to investigate textual languages as well.

In the sequel I introduce the tools, the case study and the work I am planning to do.

2. Tools

2.1. AToMPM

Throughout the course [1] we used AToMPM a lot. AToMPM [3] stands for "A Tool for Multi-Paradigm Modeling". It is a research framework for designing DSML environements, performing model transformations, and manipulating and managing models. It runs completely over the web and follows the philosophy of modeling everything explicitly, at the right level of abstraction(s), using the most appropriate formalism(s) and process(es), being completely modeled by itself.

In AToMPM we used a number of visual formalisms:

- `SimpleClassDiagrams`-formalism to specify the abstract syntax of a language,
- `ConcreteSyntax`-formalism (for concrete syntax),
- `TransformationRule`-formalism to specify model transformation rules and
- `MoTif`[1] [4] to combine these rules into programs.

Email address: `fedor.biryukov@student.uantwerpen.be` (Fedor Biryukov)

[1] `MoTif` is an acronym for the Modular Timed graph transformation formalism.

2.2. Xtext

Xtext [5] is a framework for development of programming and domain-specific languages. With Xtext you define your language using a powerful grammar language. As a result you get a full infrastructure, including parser, linker, typechecker, compiler as well as editing support for Eclipse, IntelliJ IDEA and your favorite web browser.

In Xtext, every language's concrete syntax is defined by a grammar and every abstract syntax is defined by an Ecore model. Xtext can automatically derive the Ecore model from a grammar or import an existing Ecore model. It can be said that the grammar mixes both concrete and abstract syntax definition and you may have your concerns about this. However, Eysholdt [6] reports that it proved to be very productive to start with a derived Ecore model and to stop regenerating it as soon as the language's abstract syntax is stable.

3. Case study

I will reconsider the railway system case study used throughout the course [1]. We have already solved it in metaDepth and in AToMPM. This time I will use Xtext (and possibly some other tools). Afterwards I will compare all these different approaches.

To be more concrete, I will develop a grammar for the `Railways` formalism and a grammar for the `Schedules` formalism. I will also look for a way to model the operational semantics.

Table 1: means to define the formalism syntax in different tools

Syntax	AToMPM	metaDepth	Xtext
Abstract	`SimpleClassDiagrams` model	metaDepth model	Ecore model
Concrete	`ConcreteSyntax` model	default, cannot redefine	Xtext grammar

References

[1] Model driven engineering, `http://msdl.cs.mcgill.ca/people/hv/teaching/MSBDesign/`, accessed: 2015-12-11.

[2] D. L. Moody, The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Transactions on Software Engineering 35 (6) (2009) 756–779. `doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2009.67`.

[3] Atompm, `http://www-ens.iro.umontreal.ca/~syriani/atompm/atompm.htm`, accessed: 2015-12-11.

[4] E. Syriani, H. Vangheluwe, A modular timed graph transformation language for simulation-based design, Software & Systems Modeling 12 (2) (2013) 387–414. `doi:10.1007/s10270-011-0205-0`.
URL `http://dx.doi.org/10.1007/s10270-011-0205-0`

[5] Xtext framework, `https://eclipse.org/Xtext/`, accessed: 2015-12-11.

[6] M. Eysholdt, J. Rupprecht, Migrating a large modeling environment from xml/uml to xtext/gmf, in: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, OOPSLA '10, ACM, New York, NY, USA, 2010, pp. 97–104. `doi:10.1145/1869542.1869559`.
URL `http://doi.acm.org/10.1145/1869542.1869559`