Spoofax vs Xtext

L.Elezi

leonard.elezi@student.uantwerpen.be

Abstract

Spoofax and Xtext are two language workbenches for creating external DSLs. This paper describes them briefly and gives an overview of the relevant literature and other sources of information that are needed in order to perform a comparative study of these two tools.

Keywords: Spoofax, Xtext, language workbench, DSLs

1. Introduction

Domain-specific languages (DSLs) provide high expressive power focused on a particular problem domain [1]. There are many tools that ease the development of various aspects of DSLs. This aspects include areas such as parsers for the syntax of the language, semantic analysis for constraint checking, transformation, code generation and integration of the language to other tools to make it usable.
Integrating these language construction tools under a single IDE environment coined the term language workbench. In this paper two such language workbenches will be described. The next section describes Spoofax while the third section describes Xtext. Finally the paper concludes with the schedule for the practical project that will serve as a basis for the comparative study.

2. Spoofax

The Spoofax language workbench is built on top of the Eclipse platform. It uses a text based approach to developing DSLs. Spoofax offers tools to define grammars and transform the DSL to the target language all through an IDE. One can edit and develop DSL without needing to run multiple instances of Eclipse at the same time, which aids with debugging and feedback. The Syntax Definition Formalism (SDF) is used in order to define

grammars while the Stratego transformation language is used to transform the DSL to the target language. On top of this, Spoofax provides a special editor language which helps with code-outline, code completion and syntax highlight. The Eclipse IDE Meta-tooling Platform IMP is used heavily by Spoofax. This offers the possiblity to export the final language as a regular Eclipse plug-in, which makes it easy to distribute it to other possible users.

2.1. Syntax Definition Formalism

The SDF-language is a declarative language[2]. Since it is a grammar definition language it's goal is to describe production rules of the grammar. It does not allow embedding of any code or rules with side effects. SDF is backed by a scannerless generalized-LR parser [3] which allows it to support the full range of context free grammars. This means that it can combine different grammars into one and also reuse already existing grammars.

2.2. Transformations

The Stratego transformation language is used by Spoofax te describe transformations from the Abstract Syntax Tree source onto the target language. Stratego provides writing reduction rules that describe transformation steps. These transformations are rules that first remove the syntactic sugar from the source language and then describe the transformation from the basic language building blocks to the target language. Transformations are a fundamental concept of Spoofax. Features like error-messages, auto-completion or type references can be described as special transformation of the source code. For example for auto-completion the source code is transformed into a list of possible types, methods and fields. This is how one can provide complex IDE functionalities in Spoofax.
In the case of transformations to complex errors (such as compiler error messages) Spoofax provides also the location of the error. This is done by associating each node of the AST with the origin position in the source code file. Thus the position information is preserved after the transformations are applied and when error reporting, it is sufficient to return the right marked node.

2.3. Editor language

A special editor language is used by Spoofax to define the features of the generated IDE. In this editor one can define IDE functionalities such as code-folding, syntax highlighting and positions where code completion may

be invoked. The lists generated by the Stratego transformations are used as information source in order to help with the above definitions. For example code-completion uses the list of types generated by Stratego.

2.4. Spoofax literature and other related material

There are not many papers on Spoofax but the project is active and has an online presence where there is a wealth of documentation and tutorials.

2.4.1. The Spoofax Language Workbench Rules for Declarative Specification of Languages and IDEs, Lennart C. L. Kats, Eelco Visser [4]

This is a paper from the creators of Spoofax. It describes the architecture of the tool and and the particularities of Spoofax language definitions. They discus the syntax definition, the specification of syntactic editor services and the definition of language semantics such as analysis, transformations and code generation. The authors also give a their experience with language development using Spoofax.

2.4.2. Stratego: A Language for Program Transformation based on Rewriting Strategies System Description of Stratego 0.5, Eelco Visser [5]

One important concept of Spoofax as previously mentioned is it's use of transformations through the Stratego transformation language. This paper gives a description of the Stratego system, more exactly of version 0.5. It discusses the features of the language, it's library and compiler and in the end it introduces some of the applications built with it.

2.4.3. http://strategoxt.org/Spoofax/ [6]

This is the Spoofax main website. It has a fairly good amount of documentation and tutorials on the language workbench itself and all it's building components such as Stratego.

2.5. Spoofax Setup

Since Spoofax is used through the Eclipse IDE, it is straightforward to install and setup. All is needed is the Eclipse IDE set to the Plugin Development environment. The installation is done through the Eclipse Update site mechanism. Of course the drawback is that one needs some experience with Eclipse.

3. Xtext

Xtext is a platform for developing programming languages and DSLs, by offering the developer with a set of previously built DSLs and APIs. In the end, a full implementation of the language is given that can be run on any JVM independently of the Eclipse IDE. Extext covers all aspects of a complete language infrastructure beginning from the parser to interpreter, code generator and so on. It also generates a fully featured IDE based on Eclipse.
Xtext has a tight integration with EMF which allows it to integrate with other tools and mix graphical and textual languages. The grammar of the DSL is specified with a textual language. The code generation is done with Xtend 2, a java like language. One thing noticeble is that Xtext uses Java for constraints validation. Xtext is one of the most popular and most used language workbench for creating external DSLs.

3.1. Eclipse Modelling Framework - EMF

EMF is a framework and code generation facility that allows to build Java applications based on structured data models. EMF uses Java, XML and UML in order to make modeling as practical and hassle free as possible. EMF enables fine-grained data sharing among tools and applications. It can generate Java source code that allows to CRUD and serialize/deserialize instances of models. It uses it's own simple metadata called Ecore.

3.2. Xtend

Xtend is a statistically-typed programming language which translates to comprehensible Java source code. It uses the Java language and improves on many aspects such as Extension methods, lambda expression etc. Everything written in Xtend cooperates with Java 100% and as expected.

3.3. Xtext literature and other related material

The majority of the papers on Xtext were outdated (e.x using oAW), but the main Xtext site made up for that more than enough.

3.3.1. http://www.eclipse.org/Xtext/index.html [7]

This is the main website of Xtext. It offers all the possible documentation and help. It also contains several tutorials and step by step examples.

3.4. Xtext setup

Installing Xtext is also fairly easy. They offer a full preconfigured package of Eclipse that has all the necessary plugins. All is needed is to extract and run it.

4. Other reading materials

Except papers and website on these two specific tools, several papers on comparison between other language workbenches have been studied.

4.1. A Comparison of Tool Support for Textual Domain-Specific Languages, Michael Pfeiffer, Josef Pichler [8]

This paper describes four different tools (openArchitectureWare, Meta Programming System, MontiCore, IDE Meta-Tooling Platform) supporting the creation of language workbenches. The paper was reviewed mostly to get an idea of the set of criteria that can possibly be used in our comparative study.

4.2. Comparing Language Workbenches, Roman Stoffel [9]

This paper also compares three tools (MPS, Spoofax and Actifsource). The comparison is done on the way these tools handle the definition and implementation of DSLs.

4.3. http://www.languageworkbenches.net/ [10]

This is the main website of the Language Workbench Challenge, a recurring workshop to help people get more insight on what tools that facilitate the development of DSLs are capable of.

5. Schedule for the project

First a set of comparing criteria will be chosen, probably from the above mentioned papers. Then a DSL example that can put the tools to a fair test will be developed. In the end a table with comparisons based on the selected criteria and the results from the practical example will be compiled and discussed in a paper.

6. Bibliography

[1] M. Mernik, J. Heering, and A. Sloane. When and how to develop domain-specific languages. ACM Computing Surveys (CSUR), 37(4):344, 2005.

[2] J. Heering, P. R. H. Hendriks, P. Klint and J. Rekers [1989], The syntax definition formalism sdf, reference manual, SIGPLAN Not. 24, 4375.

[3] E. Visser [1997], Scannerless generalized-lr parsing, Technical report

[4] The Spoofax Language Workbench Rules for Declarative Specification of Languages and IDEs, Lennart C. L. Kats, Eelco Visser

[5] Stratego: A Language for Program Transformation based on Rewriting Strategies System Description of Stratego 0.5, Eelco Visser

[6] `http://strategoxt.org/Spoofax/`

[7] `http://www.eclipse.org/Xtext/index.html`

[8] A Comparison of Tool Support for Textual Domain-Specific Languages, Michael Pfeiffer, Josef Pichler

[9] Comparing Language Workbenches, Roman Stoffel

[10] `http://www.languageworkbenches.net/`