Spoofax: An Extensible, Interactive Development Environment for Program Transformation with Stratego/XT

Karl Trygve Kalleberg [1]

Department of Informatics, University of Bergen, P.O. Box 7800, N-5020 BERGEN, Norway

Eelco Visser [2]

Department of Software Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands

1 Introduction

Many programmable software transformation systems are based around novel domain-specific languages (DSLs), with a long history of development and successful deployment. Despite their maturity and applicability, these systems are often discarded as esoteric research prototypes. This is partly because the languages are frequently based on less familiar programming paradigms such as term and graph rewriting or logic programming. Another reason is that modern development environments are rarely found for these systems. The basic and expected interactive development aids such as source code navigation, content completion, syntax highlighting and continuous error checking, are rarely available to developers of transformation code.

The lack of development aids keeps the entry barrier for new developers high; DSLs for program transformation use their own syntax and language constructs which are unfamiliar to many, and most editing environments support these rather poorly, providing only limited syntax highlighting, and little else. Even skilled developers are less effective, because errors are reported late in the edit-compile-run cycle, only after compiling. It is generally held that errors should be reported immediately after a change has been made, while the human programmer is still in a relevant frame of mind. Also, errors should ideally be customizable and check project-specific design rules, if possible.

Stratego/XT is a domain-specific language and toolset for developing standalone software transformation systems based on formal language descriptions.

[1] Email: karltk@ii.uib.no
[2] Email: visser@acm.org

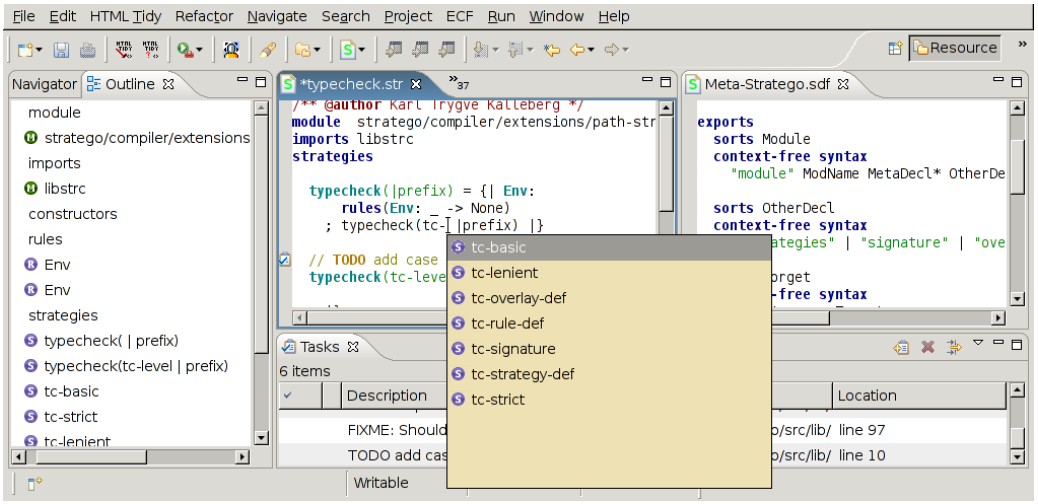

Fig. 1. Screenshot showing SDF and Stratego editors with outline view.

It is fairly mature and has been applied by various research groups and companies to tasks ranging from theorem proving to compiler implementation to domain-specific optimization to language extension, see [2]. Until recently, no good editing environment existed for Stratego, which made development harder than necessary.

In this system description paper, we describe Spoofax, an extensible, interactive environment based on Eclipse for developing program transformation systems with Stratego/XT. Spoofax supports Stratego/XT by providing modern development aids, such as customizable syntax highlighting, code outlining, content completion, source code outlining and navigation, automatic and incremental project rebuilders. The contributions of this environment include user extensibility with scripts written in Stratego that allow live analyses and transformations of the code under development; syntax highlighting, navigation and content completion that eases the learning curve for new users of Stratego; and integration into a mainstream tools platform that is familiar to developers and that runs on most desktop platforms.

2 Description

Spoofax is a set of Eclipse plugins – a Stratego and an SDF editor, a help system and a Stratego interpreter. It supplements Stratego/XT, which must be installed separately, by providing an extensible, interactive development environment. Figure 1 shows a session with an SDF editor (top right), a Stratego editor (top middle), a list of pending tasks extracted from all project files (bottom), and a code outline view (left) displaying all imports, rules and strategies defined in the edited file. The popup is the content completer showing alternatives for the `tc-` prefix. Stratego/XT programs can be compiled

and executed from within the environment.

A notable feature of Spoofax is that users can write scripts in Stratego to extend the editor. These scripts perform code transformations and project-specific style or error checking on the Stratego code under development. For example, a script may ensure that no unwanted module dependencies creep in by continously checking the import list during editing. Scripts are compiled to an abstract machine format by the Stratego compiler, and the resulting files are loaded into the editor and executed inside the environment. Execution can happen on-demand or attached to pre-defined hooks, such as whenever a file is saved. This is an attractive feature because Stratego is a mature language for language processing, and its standard library provides a formal language description and reusable transformations for Stratego. This eases the writing of language processing scripts considerably, compared to other scriptable editors like the Emacs family, as scripts in Spoofax operate on the abstract syntax tree.

3 Implementation

Stratego is a modular language. Each module is defined in a source file that contains definitions of rules and strategies; it may import other modules. Spoofax maintains an in-memory representation of all modules of a project, and their import dependencies, in what we call a *build weave*. This is used by the source-code navigator, the content-completer, and the code outliner. The module dependencies are resolved by parsing the `Makefile`s in the source tree, and extracting the module include paths defined there.

The editor is built on top of three different parsers of Stratego. The ones used for syntax highlighting and code outlining are hand-written in Java, because they must work work well for syntactically incorrect programs. A scannerless GLR parser is used to extract the abstract syntax tree from source files, and these are available for user scripts to inspect. Modification is also possible, but layout is not (yet) always properly preserved.

Spoofax comes with an interpreter, written in Java, for executing compiled Stratego scripts.

4 Related Work

Many program transformation systems provide some form of interactive environments. We briefly mention some that are advanced and actively developed.

The *Meta-Environment* is an open and extensible framework for language development, source code analysis and source code transformation based on the ASF+SDF transformation system [4]. The environment provides interactive visualisations, editors with error checking and syntax highlighting. *Tom*

is a software environment for defining transformations in Java [3] and comes with a basic Eclipse editor plugin that provides syntax highlighting, context-specific help, error checking and automatic compilation, but no source navigation. *JTransformer* is a Prolog-based query and transformation engine for Java source code, based on Eclipse. It provides a Prolog editor with syntax highlighting, auto-completion, code outlining, error checking and context-specific help. *ANTLRWorks* [1] is a graphical development environment for developing and debugging ANTLR grammars, with an impressive feature list that includes code navigation, visualisations, error checking and refactoring.

All these systems have feature sets overlapping with Spoofax, but to our knowledge, only the Meta-Environment was also designed to be extensible using a transformation language.

5 Conclusion

We have introduced an extensible, interactive development environment for Stratego/XT that provides modern development aids like content completion, source code navigation, customizable syntax highlighting, automatic and incremental project building. Users can extend the environment with scripts written in Stratego, and these can perform analysis and transformation on the code under development. We feel that our environment lowers the entry level for new users by plugging into a familiar and widely available platform, and that it makes existing developers more productive by making errors quickly visible during editing.

References

[1] J. Bovet and T. Parr. ANTLRWorks: The ANTLR GUI development environment. Home page at www.antlr.org/works/ (visited 2006-12-10).

[2] M. Bravenboer, K. T. Kalleberg, R. Vermaas, and E. Visser. Stratego/XT 0.16. Components for transformation systems. In *ACM SIGPLAN 2006 Workshop on Partial Evaluation and Program Manipulation (PEPM'06)*, Charleston, South Carolina, January 2006. ACM SIGPLAN.

[3] P.-E. Moreau, C. Ringeissen, and M. Vittek. A pattern matching compiler for multiple target languages. In *12th International Conference on Compiler Construction*, LNCS, pages 61–76. Springer, 2003.

[4] M. van den Brand, A. van Deursen, J. Heering, H. A. de Jong, M. de Jonge, T. Kuipers, P. Klint, L. Moonen, P. A. Olivier, J. Scheerder, J. J. Vinju, E. Visser, and J. Visser. The ASF+SDF meta-environment: A component-based language development environment. In R. Wilhelm, editor, *Proc of the 10th Intl Conf on Compiler Construction*, volume 2027 of *LNCS*, pages 365–370. Springer, 2001.

A Demonstration

We will demonstrate how Spoofax improves the development of language processing tools from formal language descriptions with Stratego/XT by covering the following topics:
Starting a new project – We show how to set up Spoofax and Stratego/XT and create a new project from scratch that does simple transformations on a toy language called TIL, and how to configure the environment to build your project automatically and incrementally whenever a source file is saved.
Navigating the code – We demonstrate how the source code navigation can be used to jump to other modules, definition sites of rules and strategies, how to search for modules, rules and strategies with wildcards, and how the source code outliner works.
Extending the editor – The main part of the demonstration will be devoted developing a small extension to the Stratego editor that analyses the AST of the Stratego source code whenever a file is saved, how to develop such scripts inside Spoofax, and how to compile and install scripts into the running environment. We show how the scripts can ask for the AST of a Stratego source file, and how the library for transforming Stratego programs, provided by Stratego/XT, makes processing Stratego code relatively easy, by exploiting features of Stratego such as generic traversals, rewrite rules, strategy combinators, dynamic rules and concrete syntax patterns.