

Declarative Language Definition with the **Spoofax** Language Workbench

Eelco Visser



PLDI | 'London' | June 16, 2020

Declarative Language Definition with the **Spoofax** Language Workbench

Eelco Visser

Joint work with Gabriël Konat, Hendrik van Antwerpen, Arjen Rouvoet, Jeff Smits, Jasper Denkers, Daniël Pelsmaeker, Andrew Tolmach, Casper Bach Poulsen, Eduardo Amorim, Vlad Vergu, Maarten Sijm, Ivo Wilms, Phil Mistelli, Aron Zwaan, Bram Crielaard, Max de Krieger, Bram Crielaard, Luka Miljak, Guido Wachsmuth, Sebastian Erdweg, Lennart Kats, Maartje de Jonge, Tobi Vollebregt, Anya Helene Bagge, Martin Bravenboer, Karl Kalleberg, ...

Declarative Language Definition with the **Spoofax** Language Workbench

Eelco Visser

Jasper Denkers
Eduardo Amorim
Hendrik van Antwerpen

What is Spoofax?

A tool for implementing programming languages

- Open source and freely available
- Used in education, research, and industry
- Requires a lot of software engineering to maintain

A long term research project

- Incubator for language engineering research
- Basis for implementation and evaluation of 100+ papers
- Imperfect approximation of the ideal language workbench

Language Implementations from Declarative Language Definitions

Goals of Spoofax

Language definition

- Define syntax and semantics of (domain-specific) programming languages

High-level and Understandable

- Can be used as reference documentation

Executable

- Can be used to generate tools

Declarative

- No need to understand algorithms

Multi-purpose

- Derive many/all tools from single definition

Correct by Construction

- Implementations sound wrt declarative semantics

Spoofox History

Stratego/XT [1997-2005]

- Command-line language processors with SDF + Stratego (C/Linux)

Spoofox [2006-2011]

- Eclipse (IMP) IDEs from SDF + Stratego + ESV (Java)

Spoofox 2 [2012-2020] (this tutorial)

- Spoofox-Core library with bindings for Eclipse, IntelliJ, command-line
- Meta-Languages: SDF3, Stratego, NaBL, NaBL2, Statix, DynSem, ...

Spoofox 3 [2019-...]

- Live language development based on PIE build system

Spoofax in Action

Research

- Language Engineering, Language Prototyping

Education

- Compiler Construction (MiniJava)
- Language Engineering Project (2020: Ada, C, ChocoPy, FlowSpec)

Academic Workflow Engineering

- WebDSL (researchr.org, WebLab, ...)

Industry

- Oracle Labs: Graph Analytics
- Canon: Several DSLs

Meta-languages

- **Syntax definition with SDF3**
- **Static semantics with Statix**
- Data-flow analysis with FlowSpec
- Transformation with Stratego
- Dynamic Semantics with DynSem/Dynamix
- Editor service definition with ESV

This Tutorial

Goal

- Demonstrate language development with Spoofax
- With focus on concepts behind two Spoofax meta-languages
- Using live programming, building a small (but not tiny) language

Part 1: Syntax

- Multi-purpose declarative syntax definition with SDF3

Part 2: Static Semantics

- Declarative specification of type systems with Statix

Tutorial Project on GitHub


The screenshot shows the GitHub interface for the repository 'statix-sandbox' by 'MetaBorgCube'. The repository has 6 watchers, 1 star, and 0 forks. The 'Code' tab is selected, showing the file tree for the 'master' branch. The current directory is 'statix-sandbox / london /'. The file tree lists several folders and one file, each with a description and a commit time.

File/Folder	Description	Commit Time
..		
london.example	feature freeze	11 hours ago
london.small	tests	4 days ago
london.statics	statics project for tutorial	9 hours ago
london.syntax.test	syntax tests	14 hours ago
london.syntax	syntax project for tutorial	9 hours ago
london.test	add prompt	9 hours ago
london	feature freeze	11 hours ago
more-tests/other	move legacy examples out of the way	2 days ago
tutorial.md	various improvements; in particular, reorganized qualified names synt...	18 hours ago

<https://github.com/MetaBorgCube/statix-sandbox/tree/master/london>

Websites

Documentation



Spoofox

latest

The Spoofox Language Workbench

Examples

Publications

TUTORIALS

Installing Spoofox

Creating a Language Project

Using the API

Getting Support

LANGUAGE DEFINITION REFERENCE

Language Definition with Spoofox

Abstract Syntax with ATerms

Syntax Definition with SDF3

Static Semantics with NaBL2

Static Semantics with Statix

Data-Flow Analysis with FlowSpec

Transformation with Stratego

Dynamic Semantics with DynSem

Editor Services with ESV

Language Testing with SPT

LANGUAGE DEVELOPMENT REFERENCE

Build and Develop Languages

Configure Languages

Running Languages from Command-line

Programmatic API

Developing Spoofox

RELEASES

Latest Stable Release

Development Release


Release Archive

Migration Guides

CONTRIBUTIONS

Contributions

Docs » The Spoofox Language Workbench


[Edit on GitHub](#)

The Spoofox Language Workbench

Spoofox is a platform for developing textual (domain-specific) programming languages. The platform provides the following ingredients:

- Meta-languages for high-level declarative language definition
- An interactive environment for developing languages using these meta-languages
- Code generators that produces parsers, type checkers, compilers, interpreters, and other tools from language definitions
- Generation of full-featured Eclipse editor plugins from language definitions
- Generation of full-featured IntelliJ editor plugins from language definitions (experimental)
- An API for programmatically combining the components of a language implementation

With Spoofox you can focus on the essence of language definition and ignore irrelevant implementation details.

Developing Software Languages

Spoofox supports the development of *textual* languages, but does not otherwise restrict what kind of language you develop. Spoofox has been used to develop the following kinds of languages:

Programming languages

Languages for programming computers. Implement an existing programming language to create an IDE and other tools for it, or design a new programming language.

Domain-specific languages

Languages that capture the understanding of a domain with linguistic abstractions. Design a DSL for your domain with a compiler that generates code that would be tedious and error prone to produce manually.

Scripting languages

Languages with a special run-time environment and interpreter

Work-flow languages

Languages for scheduling actions such as building the components of a software system

Configuration languages

Languages for configuring software and other systems

Data description languages

Languages for formatting data

Data modeling languages


Languages for describing data schemas

Web programming languages

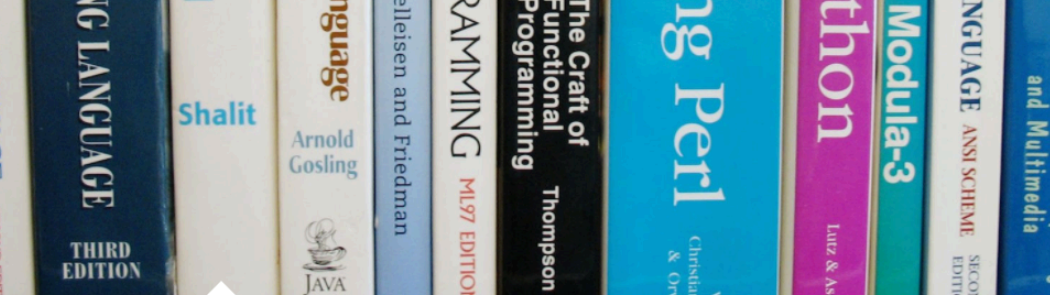
Languages for programming web clients or servers


Creating Full-Featured Editors

Compiler Construction Course



TU Delft CS4200


LECTURES
ASSIGNMENTS
PROJECT
CONTACT





Compiler Construction


Twitter


GitHub

[Edit on GitHub](#)

Compiler Construction

Course Contents

Compilers translate the source code of programs in a high-level programming language into executable (virtual) machine code. Nowadays, compilers are typically integrated into development environments providing features like syntax highlighting, content assistance, live error reporting, and continuous target code generation.

This course studies the architecture of compilers and interactive programming environments and the concepts and techniques underlying the components of that architecture. For each of the components of a compiler we study the formal theory underlying the language aspect that it covers, declarative specification languages to define compiler components, and the techniques for their implementation. The concepts and techniques are illustrated by application to small languages or language fragments.

The course covers the following topics:

- Syntax
 - concrete syntax, abstract syntax
 - context-free grammars
 - derivations, ambiguity, disambiguation, associativity, priority
 - parsing, parse trees, abstract syntax trees, terms
 - pretty-printing
 - parser generation
 - syntactic editor services
- Transformation
 - rewrite rules, rewrite strategies
 - simplification, desugaring
- Statics
 - static semantics and type checking
 - name binding, name resolution, scope graphs
 - types, type checking, type inference, subtyping
 - unification, constraints
 - semantic editor services
- Data-flow analysis
 - control-flow and data-flow
 - monotone frameworks, worklist algorithm
- Dynamics
 - dynamic semantics and code generation

Publications

Elco Visser

About

Research

Teaching

News

Blog

Contact

Publications by Year

See also: [Talks](#) | [Posters](#) | [Archives](#) | [BibTeX](#) | [Researchr](#) | [DBLP](#) | [Google Scholar](#) | [ACM DL](#) | [Researchgate](#)

2020

Constructing Hybrid Incremental Compilers for Cross-Module Extensibility with an Internal Build System

Jeff Smits, Gabriël D. P. Konat, Elco Visser.
Programming 4(3) 2020 [pdf, doi, bib, researchr]

Intrinsically-typed definitional interpreters for linear, session-typed languages

Arjen Rouvoet, Casper Bach Poulsen, Robbert Krebbers, Elco Visser.
cpp 2020 [pdf, doi, bib, researchr]

2019

Editorial Message

Elco Visser.
PACMPL 3(OOPSLA) 2019 [pdf, bib, researchr]

Fast and Safe Linguistic Abstraction for the Masses

Elco Visser.
A Research Agenda for Formal Methods in the Netherlands 2019 [bib, researchr]

From Whole Program Compilation to Incremental Compilation: A Critical Case

Jeff Smits, Gabriël Konat, Elco Visser.
Second Workshop on Incremental Computing (IC 2019) 2019 [pdf, bib, researchr]

Scopes and Frames Improve Meta-Interpreter Specialization

Vlad A. Vergu, Andrew Tolmach, Elco Visser.
ECOOP 2019 [pdf, doi, bib, researchr]

Towards Language-Parametric Semantic Editor Services Based on Declarative Type System Specifications (Brave New Idea Paper)

Daniël A. A. Pelsmaeker, Hendrik van Antwerpen, Elco Visser.
ECOOP 2019 [pdf, doi, bib, researchr]

Towards language-parametric semantic editor services based on declarative type system specifications

Daniël A. A. Pelsmaeker, Hendrik van Antwerpen, Elco Visser.
OOPSLA 2019 [pdf, doi, bib, researchr]

Precise, Efficient, and Expressive Incremental Build Scripts with PIE

Gabriël Konat, Roelof Sol, Sebastian Erdweg, Elco Visser.
Second Workshop on Incremental Computing (IC 2019) 2019 [pdf, bib, researchr]

From definitional interpreter to symbolic executor

Adrian D. Mensing, Hendrik van Antwerpen, Casper Bach Poulsen, Elco Visser.
OOPSLA 2019 [doi, bib, researchr]

A Research Agenda for Formal Methods in the Netherlands

Marieke Huisman, Wouter Swierstra, Elco Visser.
Technical report UU-CS-2019-004, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2019 [pdf, doi, bib, researchr]

2018

Towards Zero-Overhead Disambiguation of Deep Priority Conflicts

Luís Eduardo de Souza Amorim, Michael J. Steindorfer, Elco Visser.
Programming 2(3) 2018 [pdf, doi, bib, researchr, abstract]

PIE: A Domain-Specific Language for Interactive Software Development Pipelines

Gabriël Konat, Michael J. Steindorfer, Sebastian Erdweg, Elco Visser.
Programming 2(3) 2018 [pdf, doi, bib, researchr, abstract]

Scopes as types

Hendrik van Antwerpen, Casper Bach Poulsen, Arjen Rouvoet, Elco Visser.

metaborg.org

<https://tudelft-cs4200-2019.github.io/>

eelcovisser.org

#spoofax-users @ Slack

#spoofax-users 0

117 | 2 | Add a topic

Details

Today

```
test plus [[
+ 20 22
]] run compile-and-run-deb to "42
"
```

The strategy looks like this:

```
compile-and-run-deb: program -> <compile-and-run(!"true")> program
compile-and-run(ld): program -> result
  with
    <deb(!"[Compile and run] Starting compilation: ", d)> program
```

So what I expect is that the first line in my log is my AST for `+ 20 22`. This works fine when calling this strategy from my editor, but I now get the following output when running this test:

```
13:17 | INFO | stderr - {NOTE} [Compile and run] Starting compilation: "22"{TermIndex("programs.spt", 4)}
13:17 | INFO | stderr - {NOTE} [Compile and run] Starting compilation: "20"{TermIndex("programs.spt", 1)}
```

So it seems for some reason my test is calling the strategy multiple times, for multiple terms.
Does anyone know why this is happening, and how I can fix it?

47 replies

Last reply today at 14:55

Ivo Wilms

15:37

My PR with an update to the PIE language was merged into metaborg/pie/develop. I'd like to use it, where / how is it published? [repositories/snapshots](#), [repositories/public](#) and [groups/public](#) were last updated April 2019, and [repositories/releases](#) does not have pie.lang at all. (edited)

4 replies

Last reply today at 15:45

Ivo Wilms

15:52

In that same vein:

- What is the difference between <https://artifacts.metaborg.org/content/repositories/public/> and <https://artifacts.metaborg.org/content/groups/public/>?
- Does <https://artifacts.metaborg.org/content/repositories/public/> provide snapshots, releases, both or something else?

1 reply

Today at 16:08

Bram Crielaard

16:34

@Hendrik is there any documentation on the statix stratego API? I'm trying to find the length of the path between the usage of a variable and its declaration, but I can't find any documentation.

8 replies

Last reply today at 16:59

Message #spoofax-users

B

I

</>

Aa @ 😊 📎 ➡

Active community of users and developers

Questions and answers about technical issues

Send me an email for an invitation

Delft PL is Hiring

 [Delft PL](#) [Organization](#) [Education](#) [Research](#) [Events](#) [Contact](#)

Open Positions in Programming Languages

The [Delft PL](#) group is hiring!

We currently have the following open positions:

PhD Student | Dependently Typed Programming Languages
Contact: Jesper Cockx
Posted: January 2020

PhD Student / Developer | Web Programming Languages
Contact: Eelco Visser
Posted: January 2020

PhD Student | Static Semantics Specification (TUD00118)
Contact: Eelco Visser, Casper Bach Poulsen
Posted: January 2020

PhD Student | Dynamic Semantics Specification (TUD00245)
Contact: Casper Bach Poulsen, Eelco Visser
Posted: January 2020

Student Programmer | Grammar Engineering in Spoofax Team
Contact: Eelco Visser
Posted: January 2020

Student Programmer | Web Engineering in AWE Team
Contact: Eelco Visser
Posted: January 2020

2 PhD students | Software Restructuring
Contact: Eelco Visser, Casper Bach Poulsen
Posted: February 2020

Two Assistant / Associate Professors | Programming Languages (TUD00153)
Contact: Eelco Visser
Posted: April 2020

Want to learn more?

Come and do a PhD with us!

<http://pl.ewi.tudelft.nl/hiring/>