

Testing Domain-Specific Languages

Lennart C. L. Kats

Delft University of Technology
l.c.l.kats@tudelft.nl

Rob Vermaas

LogicBlox
rob.vermaas@logicblox.com

Eelco Visser

Delft University of Technology
visser@acm.org

Abstract

The Spoofox testing language provides a new approach to testing domain-specific languages as they are developed. It allows test cases to be written using fragments of the language under test, providing full IDE support for writing test cases and supporting tests for language syntax, semantics, and editor services.

Categories and Subject Descriptors D.2.5 [Software Engineering]: Testing and Debugging—Testing Tools; D.2.3 [Software Engineering]: Coding Tools and Techniques; D.2.6 [Software Engineering]: Interactive Environments

General Terms Languages, Reliability

Keywords Testing, Test-Driven Development, Language Engineering, Grammarware, Language Workbench, Domain-Specific Language, Language Embedding, Compilers, Parsers

1. Domain-specific Language Engineering

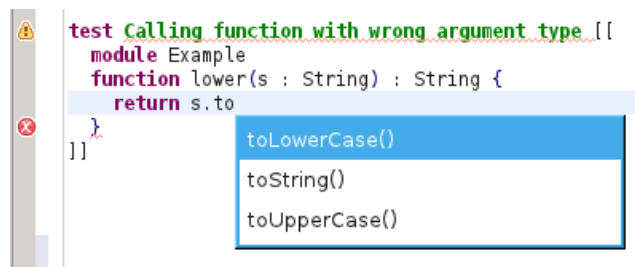
Domain-specific languages (DSLs) provide high expressive power focused on a particular problem domain. They provide linguistic abstractions and specialized syntax specifically designed for a domain, allowing developers to avoid boilerplate code and low-level implementation details.

The development of new DSLs comprises many tasks, ranging from syntax definition to code generation to the construction of an integrated development environment (IDE). The Spoofox language workbench [2] combines meta-languages for syntax definition, transformations, analyses, and editor services to form comprehensive *language definitions* that can be used to generate full interpreters, compilers, and IDE plugins.

2. Test-driven Language Development

In this demonstration we introduce the Spoofox testing language, a language-parametric testing language [1]. The testing language can be instantiated for a specific language under test, thereby integrating its syntax, semantics, and editor services into the testing language. This allows language engineers to write test cases in the language under test with full IDE support (Figure 1 (a, b)). The testing language also provides primitives for specifying assertions on tested fragments (Figure 1 (c)).

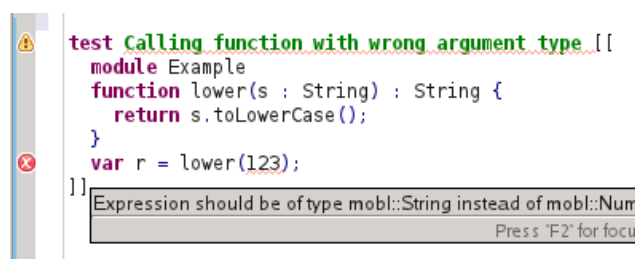
In this demonstration we show how tests can be used as the basis for an incremental, test-driven approach to language engineering. Test cases can be used to sketch and validate new syntax designs,



```
test Calling function with wrong argument type [[
  module Example
  function lower(s : String) : String {
    return s.to
  }
]]
```

toLowerCase()
toString()
toUpperCase()

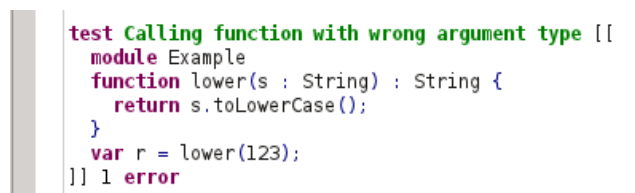
(a) Content completion for the language under test.



```
test Calling function with wrong argument type [[
  module Example
  function lower(s : String) : String {
    return s.toLowerCase();
  }
  var r = lower(123);
]]
```

Expression should be of type mobil::String instead of mobil::Num.
Press 'F2' for focus

(b) Online evaluation of tests and error markers.



```
test Calling function with wrong argument type [[
  module Example
  function lower(s : String) : String {
    return s.toLowerCase();
  }
  var r = lower(123);
]] 1 error
```

(c) Passing test case specifying negative test condition.

Figure 1. IDE support for test specifications.

through positive and negative test cases of small snippets written in the language under test. As a language definition evolves, tests can also be written for the semantics and editor services of the language under test. Test cases can check whether static semantic constraints hold (Figure 1 (c)), and can check the result of transformations and code generation. For editor services, tests can check whether hyperlinks resolve to the correct declaration, and whether services such as content completion and refactorings deliver the expected result.

Spoofox is an open source project and is publicly available at <http://spoofox.org/>.

3. About the presenters

Lennart Kats is a PhD student at Delft University of Technology, where he works on techniques and tool support for developing domain-specific languages. He is the lead developer of the Spoofox project. Rob Vermaas is a researcher and developer at Delft University and LogicBlox, and is a contributor to the Spoofox project. Eelco Visser is associate professor at Delft University, where he conducts research in the areas of language engineering, DSLs, and software deployment. He is the project lead of the TraCE, TFA, MoDSE, and PDS projects and published over 70 papers in peer-reviewed venues.

Acknowledgements This research was supported by NWO project 612.063.512, *TFA: Transformations for Abstractions* and the NIRICT LaQuSo Build Farm project.

References

- [1] L. C. L. Kats, R. Vermaas, and E. Visser. Integrated language definition testing: Enabling test-driven language development. In K. Fisher, editor, *Proceedings of the 26th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2011)*, Portland, Oregon, USA, 2011. ACM.
- [2] L. C. L. Kats and E. Visser. The Spoofox language workbench: rules for declarative specification of languages and IDEs. In W. R. Cook, S. Clarke, and M. C. Rinard, editors, *Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2010*, pages 444–463. ACM, 2010.