Integrated Language Definition Testing

Lennart Kats Delft University of Technology
Rob Vermaas LogicBlox
Eelco Visser Delft University of Technology

Language Workbenches

EntityLang.sdf

```
module EntityLang

imports Common

exports

context-free start-symbols

context-free syntax

"module" ID Definition*          ->
"entity" ID "{" Property* "}"    ->
ID ":" Type                      ->
ID                               ->
```

example.ent

```
module example

entity User {
  name     : String
  password : String
  homepage : URL
}

entity BlogPosting {
  poster : User
  body   : String
}

entity URL {
  location : String
}
```

Testing

EntityLang.sdf

```
module EntityLang

imports Common

exports

context-free start-symbols

context-free syntax

"module" ID Definition*          ->
"entity" ID "{" Property* "}"    ->
ID ":" Type                      ->
ID                               ->
```

example.ent

```
module example

entity User {
  name     : String
  password : String
  homepage : URL
}

entity BlogPosting {
  poster : Usert
  body   : String
}

entity URL {
  location : String
}
```

"DOES THE TYPE CHECKER CATCH THIS?"

Testing

EntityLang.sdf

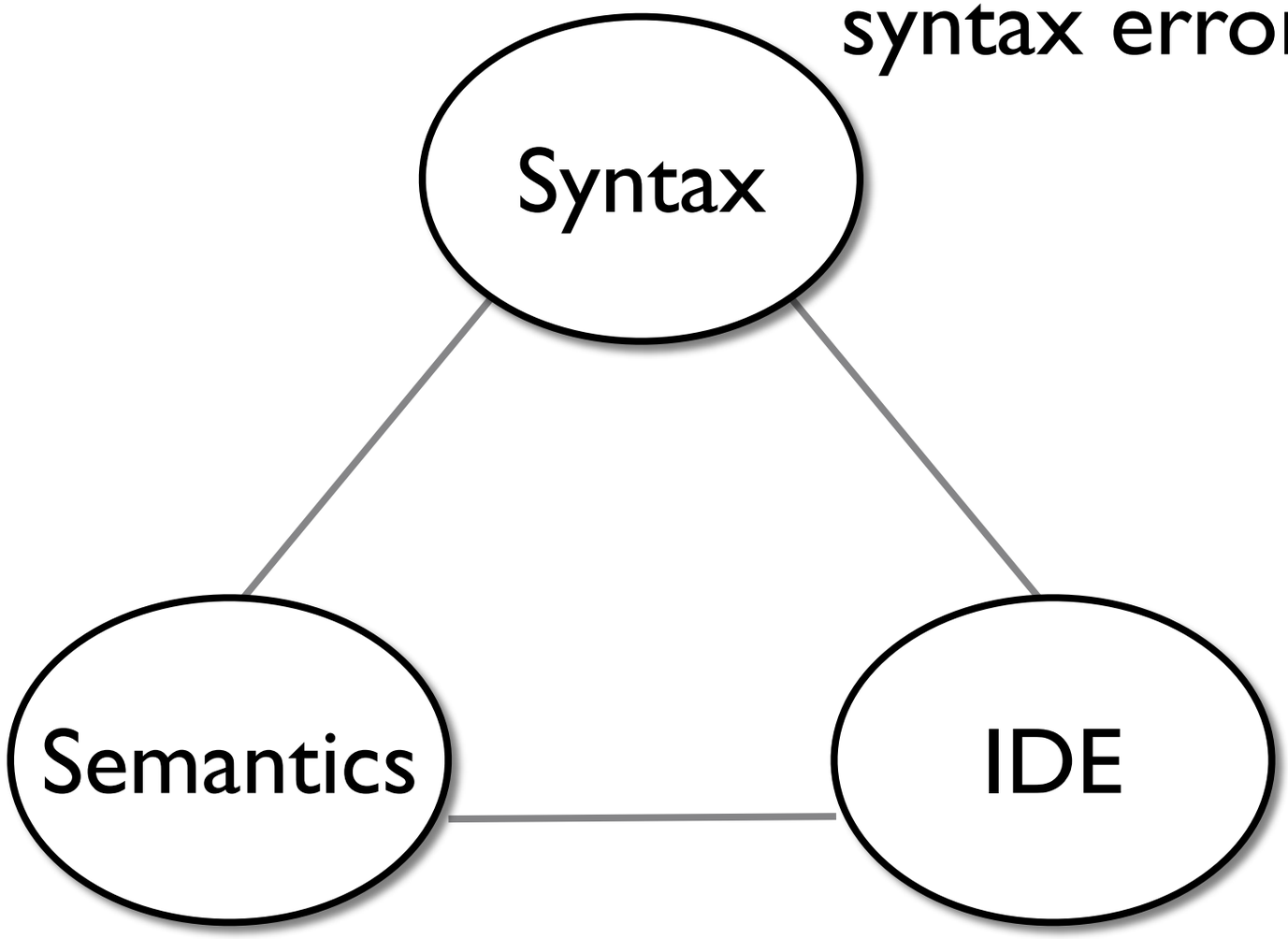
```
module EntityLang

imports Common

exports

context-free start-symbols

context-free syntax

"module" ID Definition*          ->
"entity" ID "{" Property* "}"    ->
ID ":" Type                      ->
ID                               ->
```

example.ent

```
module example

entity User {
  name     : String
  password : String
  homepage : URL
}

entity BlogPosting {
  poster : User
  body   : String
}

entity URL {
  location : String
}
```

"DOES THIS HYPERLINK POINT TO THE RIGHT PLACE?"

parsing
abstract syntax
syntax error marking



type checking
compilation
interpretation
execution

errors/warnings
reference resolving
content completion
refactoring
views

How to test language services?

IMPLEMENT FEATURE, TEST WITH EXAMPLE

DOESN'T WORK

CLOSE ENOUGH.

DISCARD TEST

How can we ***systematically*** test language definitions?

General-Purpose Testing Tools?

A Test Input

```
module Example

function foo() {
  bar();
}

function bar() {

}
```

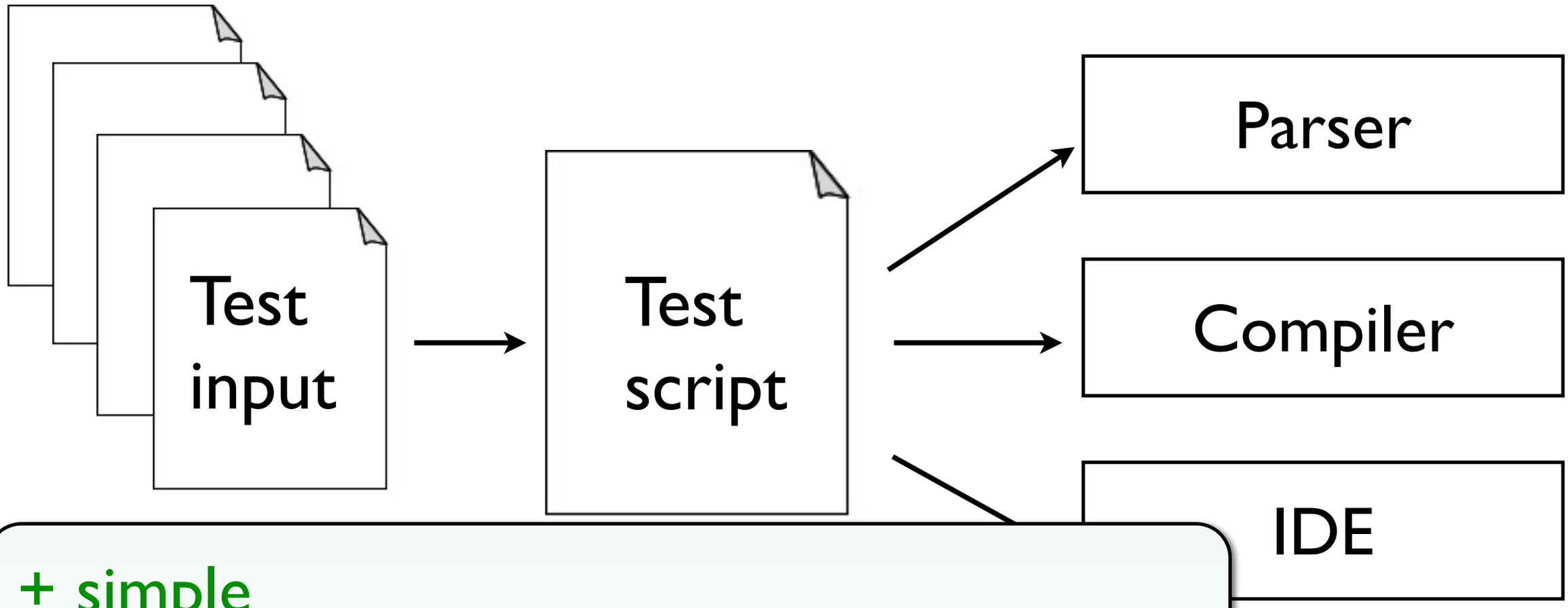
Another Test Input

```
module Example

function foo() {
  foo();
}

function bar() {

}
```

A Negative Test Case

```
module Example

function foo() {
  baz();
}

function bar() {

}
```

Automated Testing Infrastructure



+ simple
– language-specific script
– limited expressiveness
– boilerplate code
– ...

Can we design a ***general*** solution for specifying language tests?

Yes We Can

Generic test specification language

+

Parametrization

Language-Parametric Testing Language (LPTL)

```
module my-tests

language mobl

test Cannot assign an integer to a string [[
  module Example

  function test() {
    var s : String = 1;
  }
]] 1 error
```

LPTL Opportunities

Expressiveness

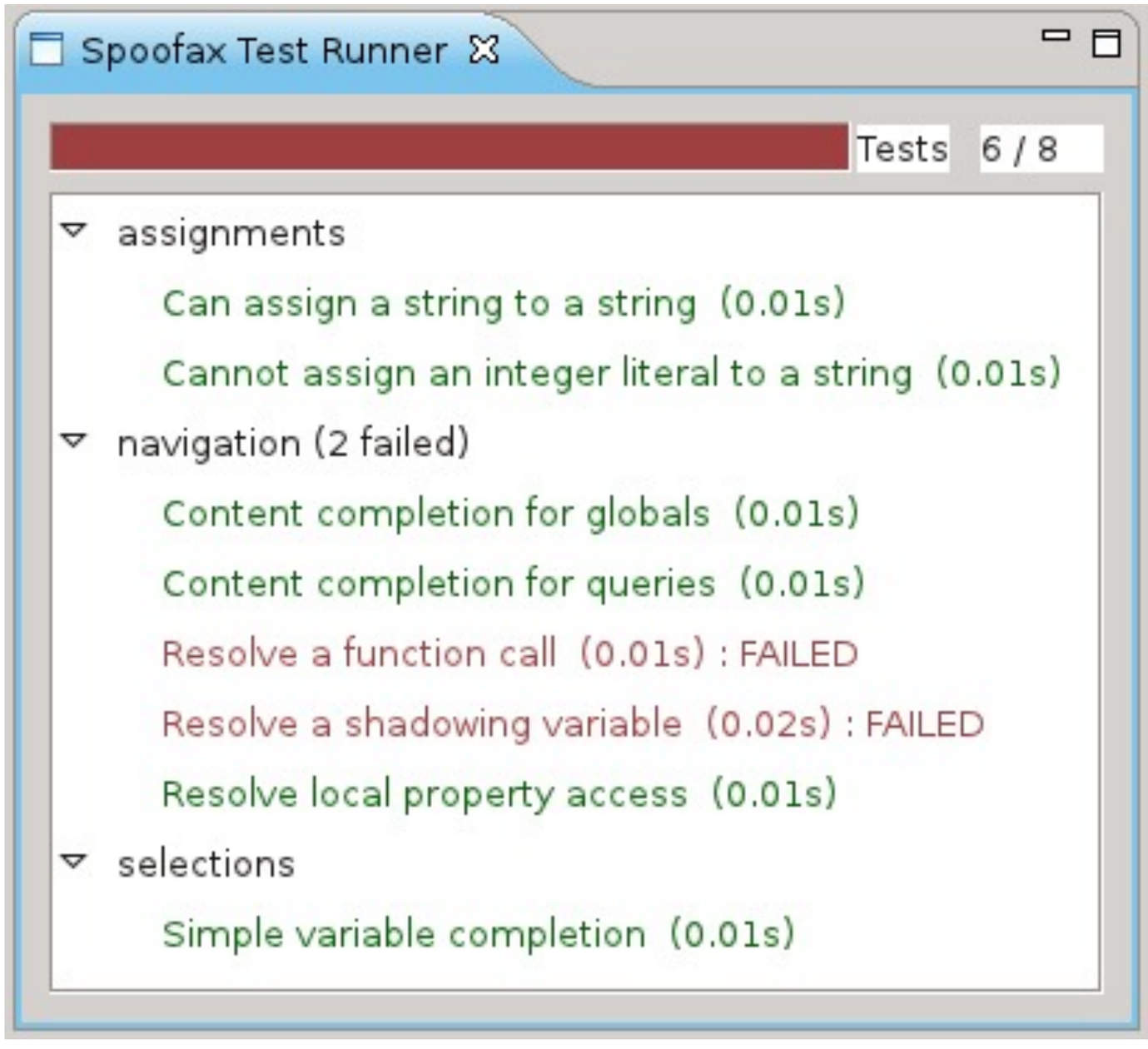
Tool support

LPTL Opportunities (1)

Module system

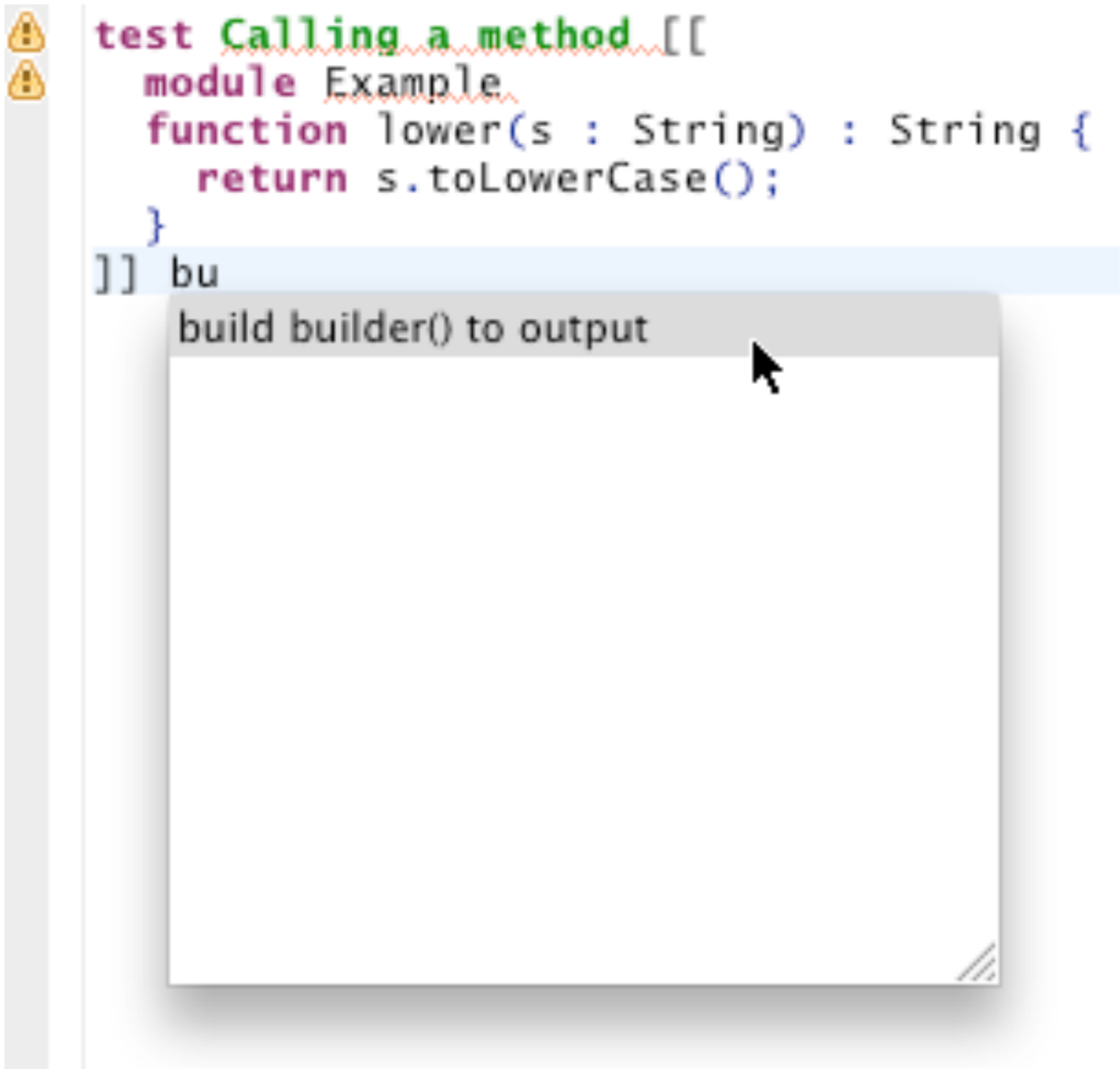
+

GUI Test Runner



LPTL Opportunities (2)

IDE support
for test
specifications



LPTL Opportunities (3)

Immediate test evaluation

*1-syntax.spt

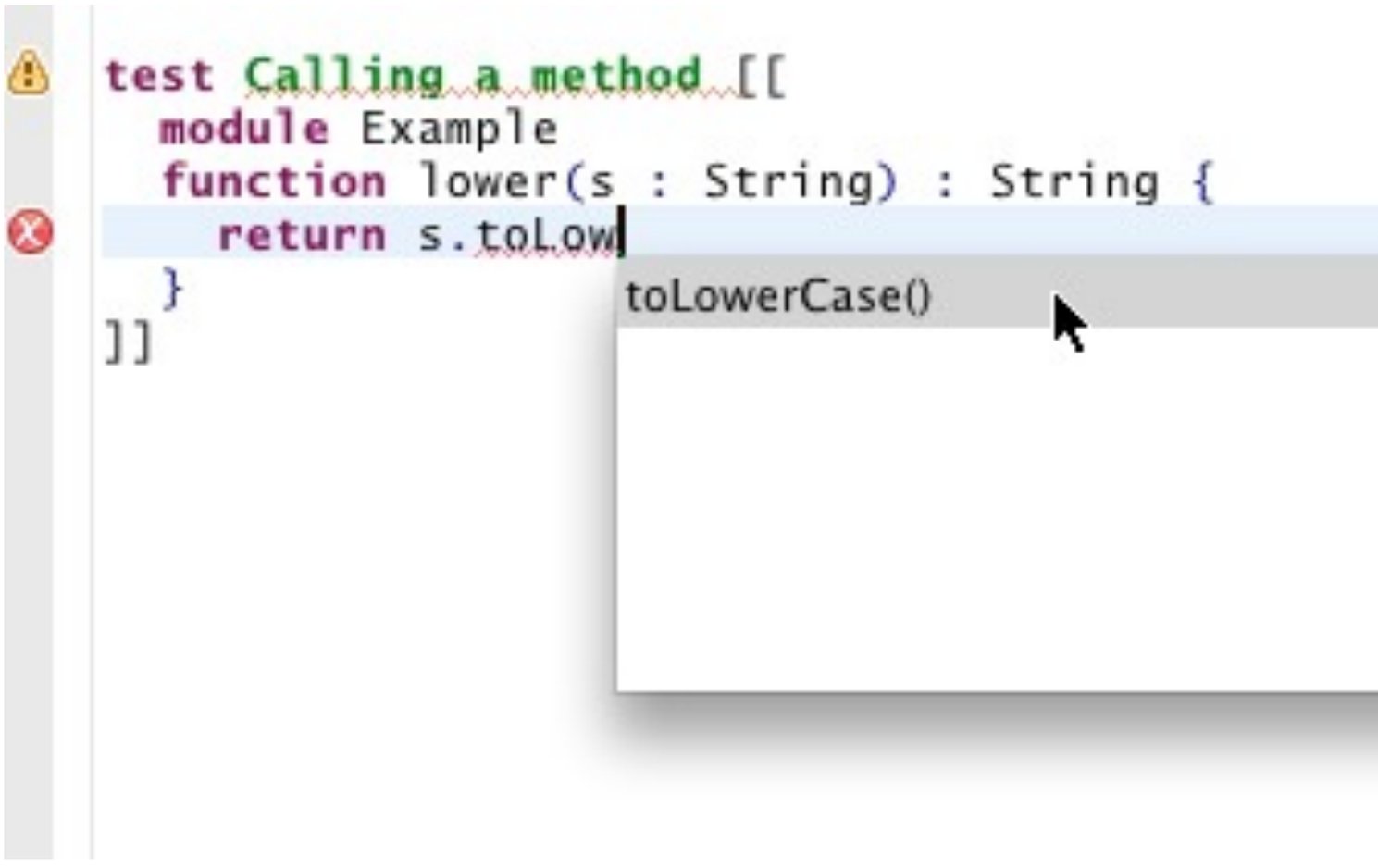
```
module syntax

language Calculang

test Add [[
  1 + 2
]] parse succeeds

test Multiply [[
  1 * 2
]]
```

Calculang.sdf

```
context-free syntax

  INT -> Exp   {cons("Int")}

  Exp "+" Exp -> Exp {cons("Plus")}
```

LPTL Opportunities (4)

IDE support for test inputs



LPTL Opportunities (5)

Reduced boilerplate

```
setup [[
  module Example

  imports stuff

  function test() {
    [[...]]
  }
]]

test Cannot assign ... [[
  var s : String = 1;
]] 1 error
```

LPTL Opportunities (6)

Wide set of
test conditions

```
1 error

2 warnings

/expected here/

parse fails

complete ... to ...

resolve ... to ...

refactor ... to ...

build ...

run ...
```

Testing Syntax (1)

```
test Proper declaration [[
  var s : String = "a";
]] parse

test Java-like declaration [[
  String s = "a";
]] parse fails
```

Testing Syntax (2)

```
test Proper declaration [[
  var s : String = "a";
]] parse to VarDecl("s", _)

test Precedence [[
  3 + 1 * 2
]] parse to [[
  3 + (1 * 2)
]]
```

Testing Error Markers

```
test Variable declaration [[
  var s : String = "a";
]] 0 errors

test Bad variable declaration [[
  var s : String = 25;
]] 1 error /wrong type/
```

Testing References

```
test [[
  module Example

  function foo() {
    [[bar]]();
  }

  function [[bar]]() {

  }
]] resolve #1 to #2
```

Testing Code Generation..?

```
test [[
  function foo() {
    return 3;
  }
]] build generate-javascript to [[
  var foo = function foo() {
    return 3;
  };
]]
```

Testing Execution

```
setup [[
  application execution

  function test() : Num {
    // init
    [[...]]
  }
]]

test Arithmetic [[
  return 1 + 1;
]] run run-test to 2
```

Implementation

Spoofax Testing Language

(spoofax.org)

Implementation Techniques

Language embedding

Dynamic instantiation of language services

Language Embedding: Syntax

```
1 module tests
2 language mobl

test Java-like declaration [[
3   String s = "a";
]] parse fails
```

Language Embedding: Semantics & IDE (1)

```
test A function call [[
  function foo() {

  }
  fo|
]]
```

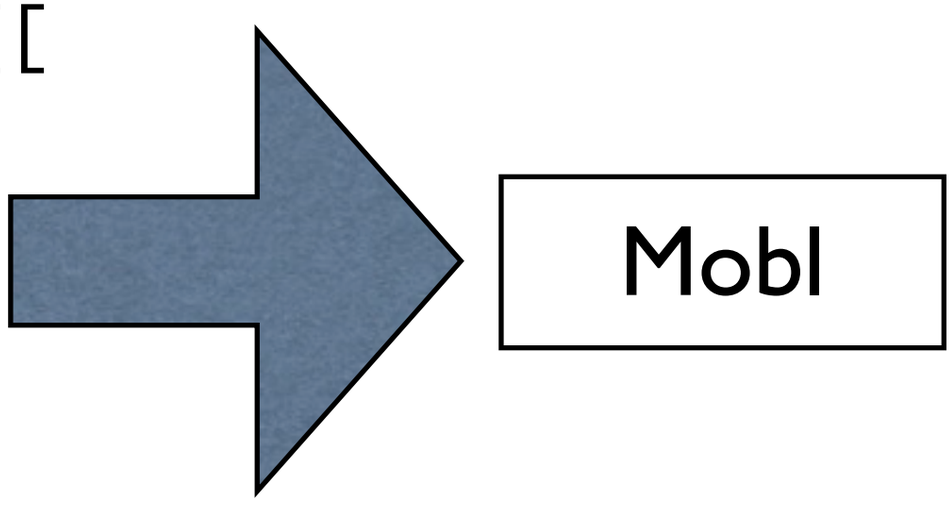
Content complete

Mobl

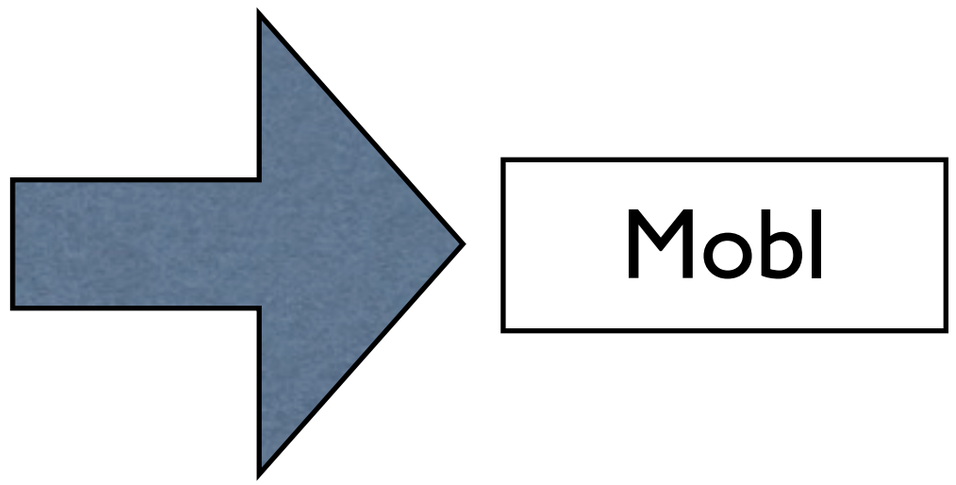
Language Embedding: Semantics & IDE (2)

```
test A function call [[
   function foo() {

   }
   notfoo();
]]
```

No condition;
error not expected



Dynamic Language Service Instantiation



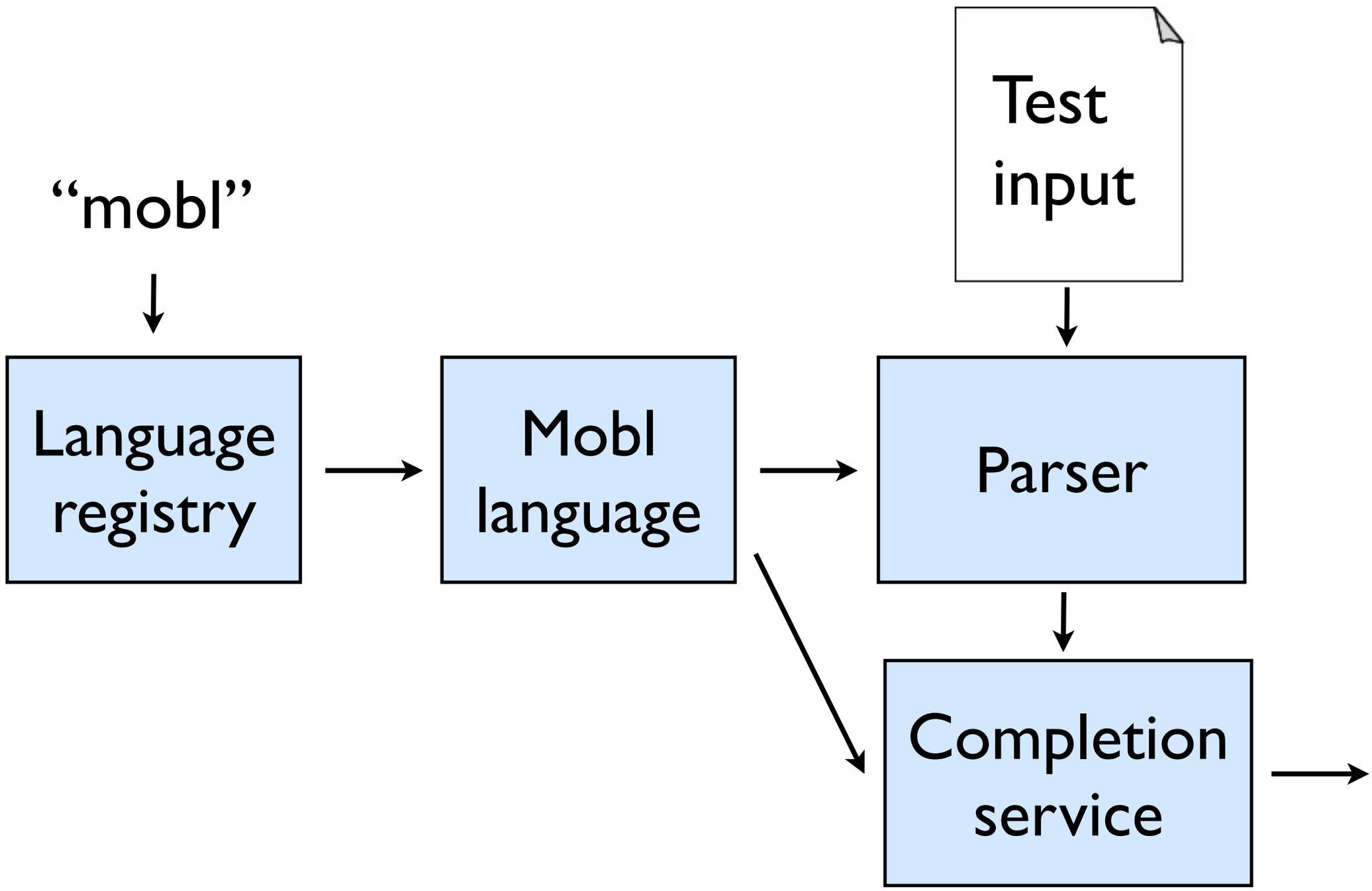
Dynamic Language Service Instantiation

Language registry

Language consists of services

Services have *functional* interfaces

Dynamic Language Service Instantiation



Reflection

+ simple
+ no scripting required
+ IDE helps avoid errors
+ little boilerplate code
+ expressiveness

Conclusions

- General abstraction for language testing
- Explored opportunities in expressiveness and tool support

TU Delft

Delft University of Technology

www.spoofax.org

LOGICBLOX®

ADDITIONAL SLIDES

Related:
Automatic Test Generation

- Generate tests from grammar
- Requires oracle
- Complementary to our approach

Self-Application

```
language Spoofax-Testing

test Testing testing [[[
    language Mobl
    test Testing [[
      module y
    ]]
]]]
```

The Spoofax Language Workbench [OOPSLA 2010]

- Integrated environment for language definition
- Define syntax, semantics, IDE
- Based on Eclipse

www.spoofax.org