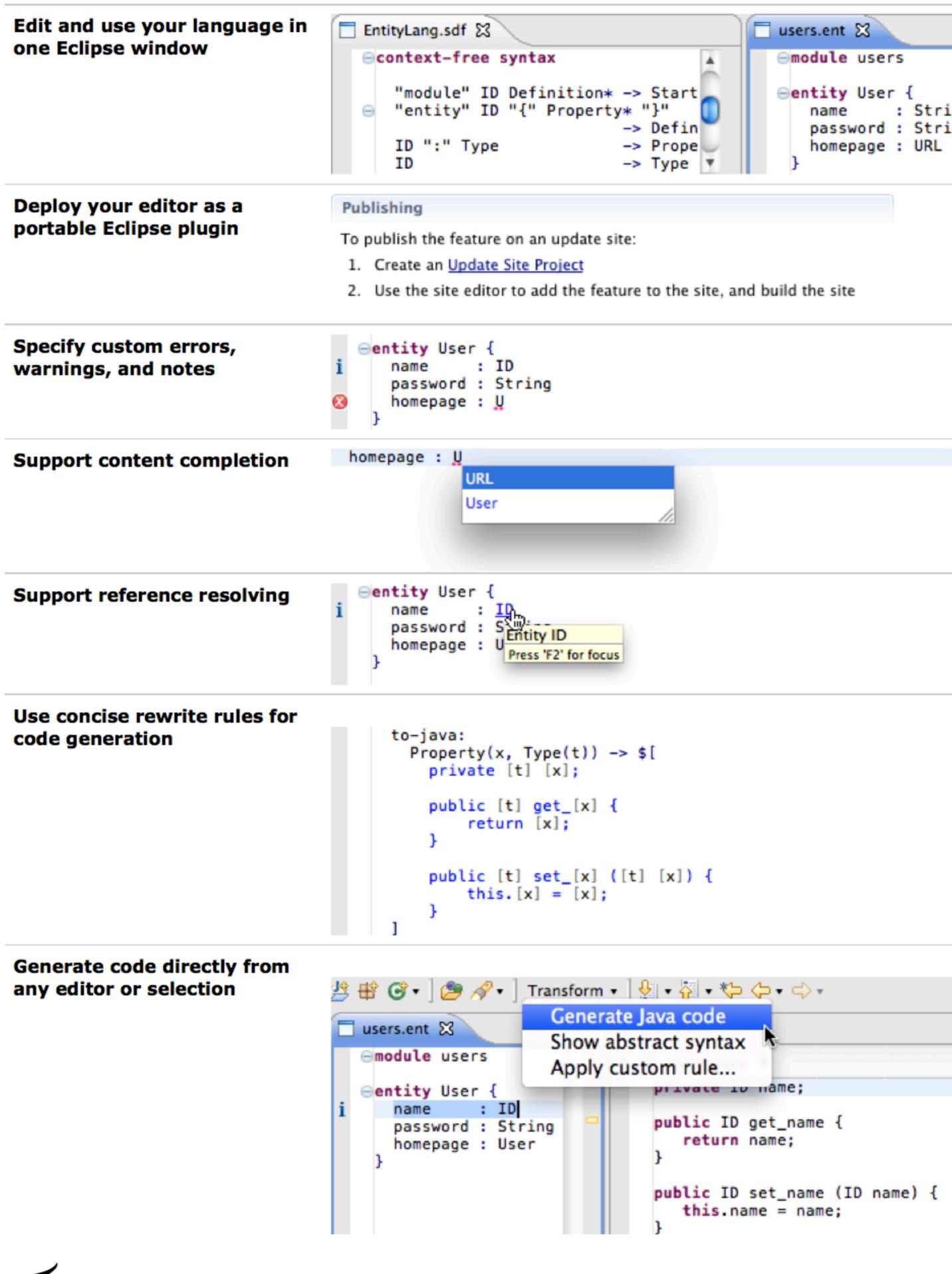
Spoofax is a *language workbench* for efficient, agile development of textual domain-specific languages with state-of-the-art IDE support. It provides a comprehensive environment that integrates syntax definition, program transformation, code generation, and declarative specification of IDE components.





Delft University of Technology

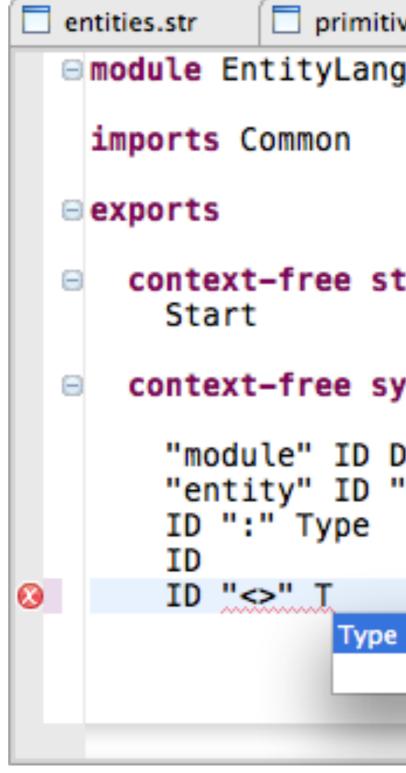
## The Spoofax Language Workbench

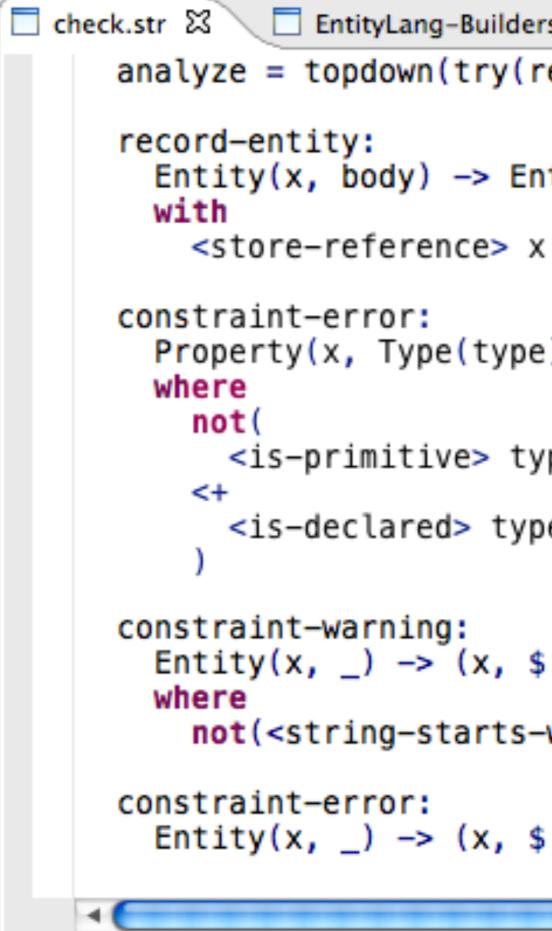
## Rules for Declarative Specification of Languages and IDEs

Lennart C. L. Kats, Eelco Visser Delft University of Technology, The Netherlands {I.c.I.kats,e.visser}@tudelft.nl

users.ent 🖾
⊖module users
<pre>entity User {     name : String     password : String     homepage : URL </pre>

• \$	 	





Use Stratego rewrite rules and strategies to define semantic editor services such as error markers, code generation, reference resolving, and content completion.

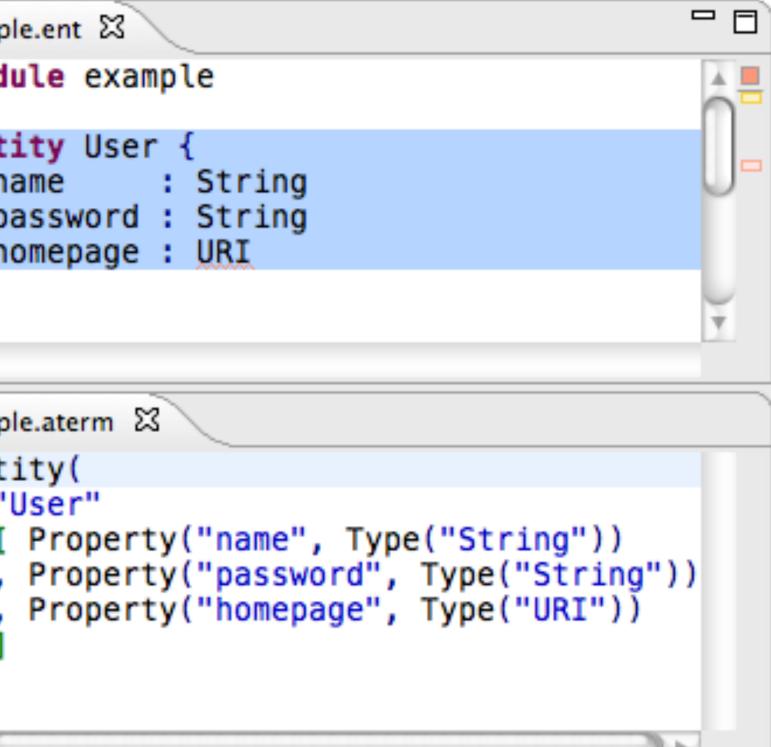
Other features of Spoofax include:

ives.str 🔲 *EntityLang.sdf 🖾 📄 Common.sdf	example
g	i ⊜modu
	⊖enti na pa 00 ho
tart-symbols	}
yntax	
<pre>Definition* -&gt; Start {cons("Module")} "{" Property* "}" -&gt; Definition {cons("Entity")} -&gt; Property {cons("Property")} -&gt; Type {cons("Type")}</pre>	□ example □ Enti □ U □ , [ ,
	4
	)
	I C

Syntax definition with modular, composable SDF grammars

EntityLang-Completions.esv 🗖 generate.str 🖾 EntityLang-Builders.esv analyze = topdown(try(record-entity)) Entity(x, body) -> Entity(x, body) Property(x, Type(type)) -> (type, \$[Type [type] is not defined]) <is-primitive> type <is-declared> type Entity(x, \_) -> (x, \$[Entity names must start with a capital]) not(<string-starts-with-capital> x) Entity(x, \_) -> (x, \$[Duplicate entity name])

- Supports the full set of context-free grammars and language composition using SDF and SGLR - Uses the Eclipse IDE Meta-tooling Platform (www.eclipse.org/imp) - Plugins created with Spoofax can be extended using pure-Java components when needed



Live abstract syntax view and custom views for transformations

```
to-java:
 Entity(x, p*) ->
 $[ class [x] {
         [p'*]
  with
   p'* := <to-java> p*
to-java:
 Property(x, Type(t)) -> $[
   private [t] [x];
   public [t] get_[x] {
       return [x];
   public void set_[x] ([t] [x]) {
       this.[x] = [x];
to-java:
 Type(t) -> t
```

www.strategoxt.org www.spoofax.org