

Domain-Specific Languages for Composable Editor Plugins

LDTA 2009, York, UK

Lennart Kats (*me*), Delft University of Technology
Karl Trygve Kalleberg, University of Bergen
Eelco Visser, Delft University of Technology

March 19, 2009

**The Framework
is the
Language**

**The IDE
is the
Language**

Implementing IDEs: The Two Faces of Eclipse

Eclipse platform:

- Cross-platform, open-source
- People *have* it
- People *use* it
 - Java, ...

Implementing IDEs: The Two Faces of Eclipse

Huge, low-level API

- SWT widgets
- synchronization
- I/O
- regexes

```
private static String[] keywords = { "module", "imports", "constructors",  
    "overlays", "where", "rules", "signature", "strategies", "sorts",  
    "if", "then", "else", "end", "let", "in", "rec", "switch", "case"};  
  
...  
public StrategoCodeScanner() {  
    ...  
  
    rules.add(new MultiLineRule("\n", "\n", string, '\\', true));  
  
    // FIXME: OMG, how this is ugly!  
    rules.add(new MultiLineRule(" ", " ", string, '\\', true));  
    rules.add(new MultiLineRule("\t", " ", string, '\\', true));  
    rules.add(new MultiLineRule("\n", " ", string, '\\', true));  
    rules.add(new MultiLineRule("(", "(", string, '\\', true));  
    rules.add(new MultiLineRule("[", "[", string, '\\', true));  
  
    rules.add(new WhitespaceRule(new StrategoWhitespaceDetector()));  
    rules.add(new StrategoDeclarationParametersRule(parameterDecl, this));  
    rules.add(new StrategoDeclarationRule(ruleDecl, ruleDecl, ruleDecl,  
        ...));  
}
```

Implementing IDEs: The Two Faces of Eclipse

Weakly typed interfaces

- XML
- java.lang.Object, IAdaptable
- CoreException

"A checked exception representing a failure."

II.

**Composable
Languages**

DSLs and Language Extensions

Domain-specific

- Database queries
- Regular expressions
- XML processing
- Matrices
- Complex numbers
- ...

```
void browse() {  
    List<Book> books =  
        <| SELECT *  
          FROM books  
          WHERE price < 100.00  
        |>;  
  
    ...  
  
    for (int i = 0; i < books.size(); i++)  
        books.get(i).title =~ s/^The //;  
}
```


Meta-programming with Concrete Object Syntax

- Program transformation
- Stratego with WebDSL, Java, XML

webdsl-action-to-java-method:

```
[[ action x_action(farg*) { stat* } ]] ->
```

```
[[ public void x_action(param*) { bstm* } ]]
```

```
with param* := <map(action-arg-to-java)> farg*;
```

```
    bstm* := <statements-to-java> stat*
```

III.

**Introducing
Spoofax/IMIP**

IDE development environments

(Or: How to Learn to Stop Worrying and Love Eclipse)

- Abstraction
 - avoid Eclipse framework complexity
- Modularity
 - separation of concerns
 - reuse
- Extensibility and customization
 - integration with existing compilers, tools

Introducing Spoofox/IMP

- Three pillars:
 - SDF grammars
 - DSLs for service descriptors
 - Implemented using Spoofox and IMP frameworks (“SAFARI”)

Spoofox



An IDE plugin created with Spoofox/IMP

The screenshot shows an IDE interface with several tabs at the top: `init.app`, `menustyle.app`, `search.app`, `administration.app`, and `author.app`. The main editor displays the following code:

```
module author

imports []

section publication list

define page publicationsTagged(author : Author, tag : Tag) {
  title { "Publications for " output(author.fullname) " tagged " output(tag.name) }
  profilePage[author, "Publications tagged " + tag.name]

  var pubs : List<Publication> :=
    select p from Publication as p, Author as a
    where (a = ~auctor) and (a in p._authors) and (~tag in p._tags);

  define body() {
    section {
      section {
        header { "Publications about " output(tag.name) }
        navigate(tag(tag)) { "All publications tagged " output(tag.name) }
      }
    }
  }
}
```

The Outline view on the right shows a tree structure:

- author
 - publication list
 - publicationsTagged
 - body
 - publicationsAll
 - body
 - profile
 - helpers

tag.name :: String
Press 'F2' for focus

Problems 6 errors, 4 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (6 items)				
Variable auctor not defined	author.app	publications	line 16	Problem
expression 42 cannot be assigned to author.fullname due to type-incompatibility.	author.app	publications	line 37	Problem
invalid left-hand side in assignment: author.name	author.app	publications	line 156	Problem

SDF and SGLR (1)

- Unified lexical and context-free syntax

```
module WebDSL
imports MixHQL[HQL] AccessControl ...
exports
  lexical syntax
    [a-zA-Z][a-zA-Z0-9\_]* → Id
    ...
  context-free syntax
    "module" Id Section*           → Unit      {cons("Module")}
    "section" SectionName Def*     → Section  {cons("Section")}
    "define" Mod* Id "{" Element* "}" → Def      {cons("SimpleDef")}
    ...
```

















SDF and SGLR (2)

- Scannerless Generalized-LR Parsing
 - Full class of context-free grammars
 - Compositional
 - Declarative disambiguation filters

```
module Stratego-WebDSL-Java-XML
imports
  Stratego-Java-15
  Stratego-WebDSL
  Stratego-XML
```

Modular Editor Service Definitions

- Main file
- Definition for each service
- Generated definitions

-  WebDSL-Analysis.esv
-  WebDSL-Analysis.generated.esv
-  WebDSL-Colorer.esv
-  WebDSL-Colorer.generated.esv
-  WebDSL-Compiler.esv
-  WebDSL-Folding.esv
-  WebDSL-Folding.generated.esv
-  WebDSL-Occurrences.esv
-  WebDSL-Occurrences.generated.esv
-  WebDSL-Outliner.esv
-  WebDSL-Outliner.generated.esv
-  WebDSL-References.esv
-  WebDSL-References.generated.esv
-  WebDSL-Syntax.esv
-  WebDSL-Syntax.generated.esv
-  WebDSL.main.esv

Reuse and Modularity in IDE plugins

*Stratego + WebDSL editor =
StrategoWebDSL editor*

- Declarative specifications
- (Backdoor available)

Creating a brand new IDE plugin

Requires:

- Syntax definition
- Language name
- File extension(s)

Gives you:

- Service templates
- Generated services
- plugin.xml, ...

And:

- Basic IDE functionality:
Coloring, outline, folding

In the Beginning: WebDSL.main.esv

module WebDSL.main

imports

WebDSL-Analysis WebDSL-Colorer WebDSL-...

language *Description*

name : WebDSL

aliases : WebDiesel

id : org.strategoxt.imp.generated.webdsl

description : "Spoofax/IMP-generated editor for the WebDSL language"

url : <http://strategoxt.org>

language *Files and parsing*

[...]

In the Beginning: WebDSL.main.esv

```
module WebDSL.main
```

```
imports
```

```
  WebDSL-Analysis WebDSL-Colorer WebDSL-...
```

```
language Description
```

```
[...]
```

```
language Files and parsing
```

```
extensions      : app
```

```
table           : include/WebDSL.tbl
```

```
start symbols  : Unit
```

In the Beginning (2): Generated Services

- Based on heuristics
- Rapid prototyping
- Starting point
 - functioning as an example
 - self-documenting

```
module WebDSL-Colorer.generated

// ...documentation...

colorer Default highlighting rules
  keyword : "Keywords" = magenta bold
  string   : "Strings"   = blue
  number   : "Numbers"   = darkgreen
  ...
```

Example: The colorer service

rules

```
webdsl-action-to-java-bean:
  [[ action x_action(farg*) {stat*} ]]
  [[ package pkgname;

    import pkgname2.*;

    @Stateful @Name("~x_actionBean")
    public class x_ActionBean implement

      @Logger private Log log = initLog

      RuleManager rules;

      @PersistenceContext(type = EXTEND
      private EntityManager entityManag

    public String x_action() {

      @Remove @Destroy
      public void destroy() {}

    }
  ]]
  where pkgname      := <BeanPackage>;
         pkgname2    := <DomainPackage>
         bstm*       := <statements-to-
```

module Stratego-WebDSL-Colorer

imports

Stratego
WebDSL

colorer *Variables*

_.Var : green **italic**

colorer *Concrete syntax*

environment *_.ToMetaExpr:*
_ gray

environment *_.FromMetaExpr:*
_ white

Example: The folding service

rules

```
webdsl-action-to-java-bean:
  [[ action x_action(farg*) {stat*} ]]
  [[ package pkgname;

    import pkgname2.*;

    @Stateful @Name("~x_actionBean")
    public class x_ActionBean implement

      @Logger private Log log = initLog

      RuleManager rules;

      @PersistenceContext(type = EXTEND
      private EntityManager entityManag

    public String x_action() {

      @Remove @Destroy
      public void destroy() {}

    }
  ]]
  where pkgname      := <BeanPackage>;
         pkgname2    := <DomainPackage>
         bstm*       := <statements-to-
```

module Java-Folding

imports

Java-Folding.generated

folding *Customization*

CompilationUnit

NewInstance

QNewInstance

ImportDec* **(folded)**

Block **(disable)**

Syntactic Editor Services

- Syntax errors
- Code folding
- Outline view
- Brace matching
- Comments
- Source code formatting

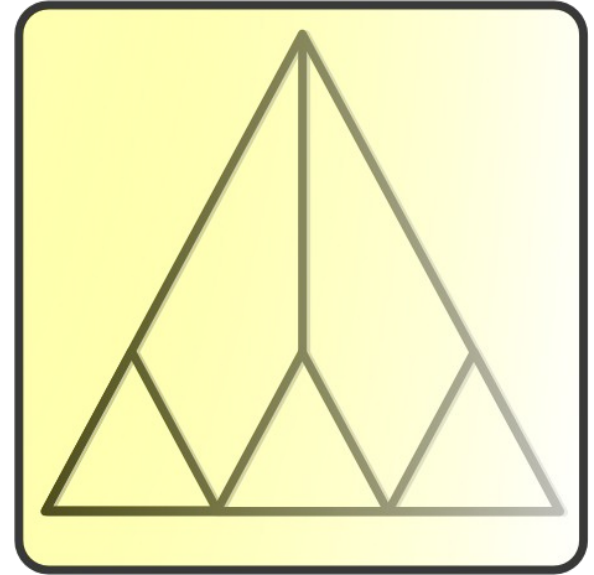
Semantic Editor Services

- Error reporting
- Reference resolving
- Reference info
- Occurrence highlighting

Stratego integration

Stratego:

- Rewrite rules
- Strategies to control their application
- Used for e.g., WebDSL, Stratego, Java [OOPSLA'08]



Interfacing with Stratego

- Interface based on rewrite rules
- Adapted primitives for parsing, caching

Offending term + message tuples

```
[(Var("auhtor"), "undeclared"), ...]
```

editor-analyze:

(ast, path, fullpath) -> (errors, warnings, infos)

with

...

(errors, warnings, infos) := <collect-all-markers> ast

...

Interfacing with Stratego

- Interface based on rewrite rules
- Adapted primitives for parsing, caching

Referenced declaration

Property("author", ...)

reference-resolve:

(ast, path, fullpath, reference) -> declaration

with

...

declaration := <find-decl> reference

...

Using Stratego: A Global-to-Local Transformation

```
entity User {  
  username :: String (id)  
  password :: Secret  
  isAdmin  :: Bool  
}
```

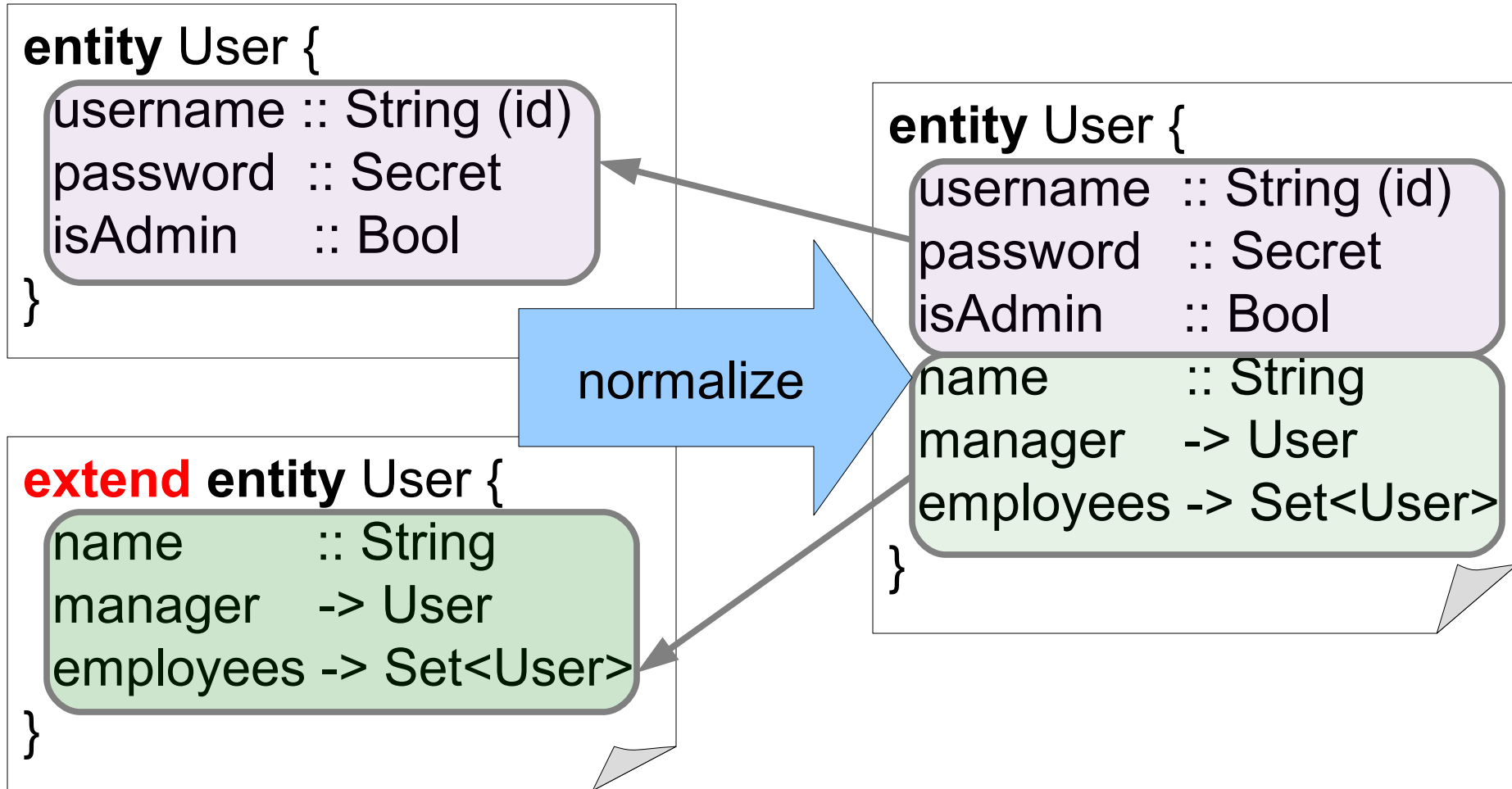
```
extend entity User {  
  name      :: String  
  manager   -> User  
  employees -> Set<User>  
}
```

normalize

```
entity User {  
  username :: String (id)  
  password :: Secret  
  isAdmin  :: Bool  
  name     :: String  
  manager  -> User  
  employees -> Set<User>  
}
```

Term Rewriting with Origin Tracking

[Van Deursen et al 1993]



Program Object Model (POM) adapter

[Kalleberg et al, LDTA'07]

Interpret term operations as API calls

- Using Spoofax interpreter
- Intercept applications of rewrite rules in strategies
- Override term building, 'all', 'some', and 'one' for origin tracking

The Ubiquitous Eclipse

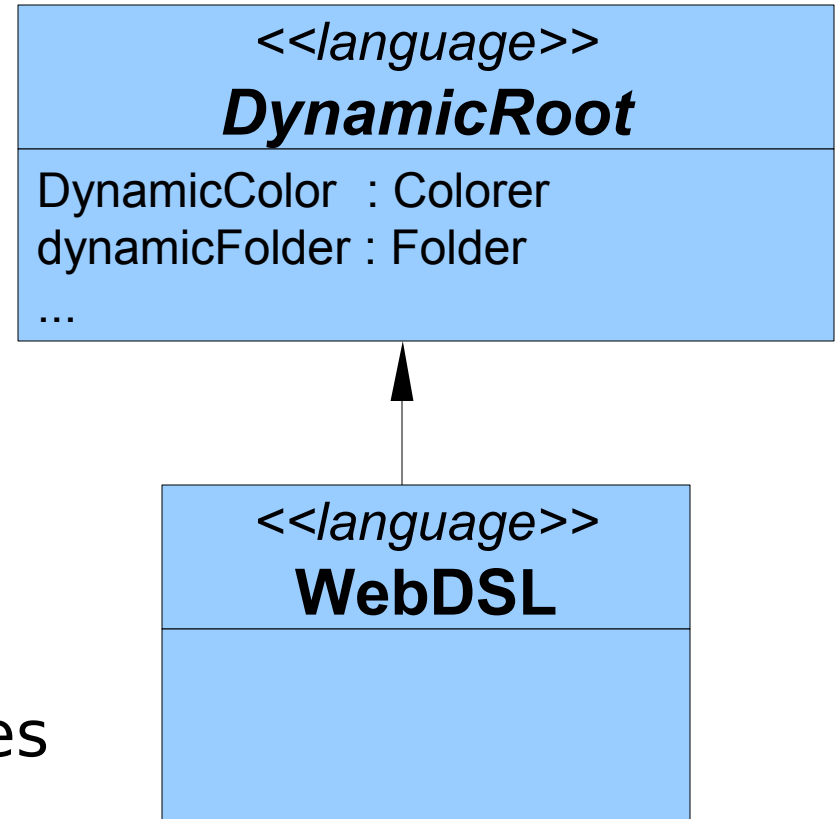


Dynamic Building and Loading: The Editor Service Builder

- Verifies all service descriptors
 - Syntax
 - Existing sorts
 - ...
- Updates plugin.xml, etc.
- Builds parse tables

Dynamic Building and Loading: Dynamically Loading Editor Services

- IMP provides:
 - static, XML-based editor declarations
 - language inheritance
- Base language
 - Defines default services
 - May be overridden by dynamic or “backdoor” implementations



Looking back

LDTA'07: Spoofax

rules

webdsl action to java bean:

```
[ action x_action(farg*) {stat* } ]| ->
|[ package pkgname;

import pkgname2.*;
import static javax.persistence.PersistenceContext
import static org.jboss.seam.ScopeType.CONVERSATION;
import static org.jboss.seam.ScopeType.SESSION;
// (...)

@Stateful @Name("~x_actionBean")
public class x_ActionBean implements x_Action {

    @Logger private Log log = initLog();

    RuleManager rules;

    // (...)
}
]|
```

Embedded Java code

```
where pkgname      := <BeanPackage>;
      pkgname2     := <DomainPackage>;
      bstm*        := <statements-to-java> stm*;
      x_Action     := <conc-strings> (<capitalize-st
      x_ActionBean := <conc-strings> (x Action "Real
```

module
imports
constructors
rules
strategies

Looking back

LDTA'08: sdf2imp

No:

- Semantic services
- Dynamic loading
- Modular definitions

rules

```
webdsl-action-to-java-bean:
  [[ action x_action(farg*) {stat* } ]] ->
  [[ package pkgname;

     import pkgname2.*;

     @Stateful @Name("~x_actionBean")
     public class x_ActionBean implements x_Action {

         @Logger private Log log = initLog();

         RuleManager rules;

         @PersistenceContext(type = EXTENDED)
         private EntityManager entityManager;

         public String x_action() {

             @Remove @Destroy
             public void destroy() {}

         }

     }
  ]]
  where pkgname      := <BeanPackage>;
         pkgname2    := <DomainPackage>;
         bstm*       := <statements-to-java> stm*;
```

Looking forward (to)

- Complete Stratego-based DSL environment
 - compiler for Java
 - SDF bundle
- Expansion of editor services
 - e.g. content completion

Looking forward (to)

- Integration with Aster [CC 2009]
- Better interactive parser
 - performance
 - error handling
 - content completion

Concluding Remarks

- Declarative DSLs
 - Avoid Eclipse API complexity
 - Specialized syntax
 - Compositionality
 - Co-evolution of language and IDE

Domain-Specific Languages for Composable Editor Plugins.
Lennart C. L. Kats, Karl T. Kalleberg, and Eelco Visser. *LDTA 2009*.

<http://www.strategoxt.org/Stratego/Spoofax-IMP>