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

```java
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("\\", "\\", string, '\\', true));
  //FIXME: OMG, how this is ugly!
  rules.add(new MultiLineRule("\r", "\r", string, '\\', true));
  rules.add(new MultiLineRule("\n", "\n", string, '\\', true));
  rules.add(new MultiLineRule("\t", "\t", string, '\\', true));
  rules.add(new MultiLineRule("\", "\", 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, this));
}
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
• ...

```java
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

```plaintext
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/IMP
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 Spoofax/IMP

• Three pillars:
  • SDF grammars
  • DSLs for service descriptors
  • Implemented using Spoofax and IMP frameworks ("SAFARI")
An IDE plugin created with Spoofax/IMP
SDF and SGLR (1)

- Unified lexical and context-free syntax

```plaintext
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
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**

```plaintext
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

```
module Stratego-WebDSL-Colorer

imports
  Stratego
  WebDSL

colorer Variables

  _\text{Var} : \text{green italic}

colorer Concrete syntax

environment _\text{.ToMetaExpr}:
  _\text{gray}

environment _\text{.FromMetaExpr}:
  _\text{white}
```

```
webdsl-action-to-java-bean:
  [[\text{action} \text{x\_action}(\text{farg}\*) \{\text{stat}\* \}}]]
  [[\text{package} \text{pkpname};]]

  import \text{pkpname2}.*;

  @Stateful @Name("\text{~x\_actionBean}")
  public class \text{x\_ActionBean} implement
    @Logger private Log log = initLog
    RuleManager rules;

  @PersistenceContext(type = EXTEND
  private EntityManager entityManage

  public String x\_action() {

    @Remove @Destroy
    public void destroy() {

  }

  where pkpname := <BeanPackage>;
  pkpname2 := <DomainPackage>
  bstm* := <statements-to-
```
Example: The folding service

```java
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

```plaintext
reference-resolve:
  (ast, path, fullpath, reference) -> declaration
  with
  ...
  declaration := <find-decl> reference
  ...

Referenced declaration

Property("author", ...)
```
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>
}

**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]

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

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

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

<<language>>

DynamicRoot

DynamicColor : Colorer
dynamicFolder : Folder
...

<<language>>

WebDSL
Looking back

LDTA'07: Spoofax

rules

```webdsl
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();
    } // (...)

where pkgname := <BeanPackage>;
    pkgname2 := <DomainPackage>;
    stmt* := <statements-to-java> stmt*;
    x_Action := <conc-strings> ( <capitalize-st
    x_ActionBean := <conc-strings> ( x_Action, "Bean";
```
Looking back

LDTA'08: sdf2imp

No:
- Semantic services
- Dynamic loading
- Modular definitions
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.

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