Demanding First-Class Equality for Domain Specific Aspect Languages

Arik Hadas
Dept. of Mathematics and Computer Science
The Open University of Israel

Joint Work With:
David H. Lorenz
Domain Specific Aspect Languages

- DSLs
- DSALs
- AOP

- COOL
- KALA
- RIDL
- AspectGrid
- AO4SQL
- Racer
DSALs are Second-class

- Second-class DSLs
  - Not as easy to develop and to use
- Second-class AOP languages
  - Incompatible with AOP tools
Background

2005

DSL

Language workbenches: The killer-app for domain specific languages

DSAL

abc: an extensible AspectJ compiler
Language workbenches: The killer-app for domain specific languages

abc: an extensible AspectJ compiler

2005
<table>
<thead>
<tr>
<th>DSL</th>
<th>Since 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>Since 2005</td>
</tr>
<tr>
<td>Language workbenches: The killer-app for domain specific languages</td>
<td>Language workbenches (LW)</td>
</tr>
<tr>
<td>abc: an extensible AspectJ compiler</td>
<td>AOP composition frameworks (CF)</td>
</tr>
<tr>
<td>Since 2005</td>
<td>Reflex, XAspects, AWESOME</td>
</tr>
</tbody>
</table>
### Background

<table>
<thead>
<tr>
<th>DSL</th>
<th>2005</th>
<th>DSAL</th>
<th>Since 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Language workbenches: The killer-app for domain specific languages</td>
<td>abc: an extensible AspectJ compiler</td>
<td>Language workbenches (LW)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– xtext, MPS, Spoofax</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AOP composition frameworks (CF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Reflex, XAspects, AWESOME</td>
</tr>
</tbody>
</table>
### Background

<table>
<thead>
<tr>
<th>DSL</th>
<th>2005</th>
<th>Since 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language workbenches: The killer-app for domain specific languages</td>
<td>Language workbenches (LW)</td>
<td>- xtext, MPS, Spoofax</td>
</tr>
<tr>
<td>abc: an extensible AspectJ compiler</td>
<td>AOP composition frameworks (CF)</td>
<td>- Reflex, XAspects, AWESOME</td>
</tr>
</tbody>
</table>

**But no workbench solution for DSALs**
Language Workbench (LW)

- DSL Code
- Java
- Bytecode

javac
Language Workbench (LW)

- DSL Code

Language Workbench

Java

javac

Bytecode
Using LW for DSAL

DSL Code

DSAL Code

Language Workbench

Java

javac

Bytecode

No Aspects
Using LW for DSAL

DSL Code
DSAL Code

Language Workbench

Java

javac

Bytecode

No Aspects
LW for AspectJ

Language Workbench

DSAL Code

Java AspectJ

javac ajc

Bytecode

Woven Bytecode

No Multiple DSALs
DSAL Complexity

- Unlike DSL, DSAL tackles crosscutting concerns
Looking for a DSAL Workbench

<table>
<thead>
<tr>
<th>Tools for creation &amp; usage of languages</th>
<th>Language Workbench</th>
<th>AOP Composition Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining weaving semantics needed for DSALs</td>
<td>☹️</td>
<td>☑️</td>
</tr>
</tbody>
</table>
Looking for a DSAL Workbench

<table>
<thead>
<tr>
<th>Tools for creation &amp; usage of languages</th>
<th>Language Workbench</th>
<th>AOP Composition Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining weaving semantics needed for DSALs</td>
<td>😞</td>
<td>😊</td>
</tr>
</tbody>
</table>

Will a naive combination of the two be a proper solution?
Naive Combination of LW and CF

DSAL Code

Language Workbench

AspectJ

\texttt{ajc} Composition Framework

Woven Bytecode

First-class DSL
But Still Second-class AOP
Traditional LW Architecture

DSAL Code

Language Workbench

Code Transformation

Composition Framework

AspectJ

Woven Bytecode
DSAL Workbench Architecture

DSAL Code

Composition Framework

Language Workbench

AspectJ DSAL Code

Code Transformation

Composition Framework

WovenBytecode

First-class AOP
DSAL Workbench Architecture

- Standalone DSAL compiler
- Can generate debugging & browsing information

Code Transformation

Composition Framework

DSAL Code

Woven Bytecode

First-class AOP
Implementation

DSAL Code

Spoofax

Code Transformation

AWESOME*

Woven Bytecode
package base;

public class BoundedStack implements Stack {
    protected Object[] buffer;
    private int usedSlots = 0;

    public BoundedStack(int capacity) {
        this.buffer = new Object[capacity];
    }

    public Object pop() {
        Object result = buffer[usedSlots - 1];
        usedSlots--;
        buffer[usedSlots] = null;
        return result;
    }

    public void push(Object obj) {
        // Multiple markers at this line
        - implements base.Stack.push
        - advised by injar aspect: BoundedStackCoord.cool
    }
}

package base;

coordinate base.BoundedStack {
    selfex {push(java.lang.Object), pop()};
    mutex {push(java.lang.Object), pop()};

    condition full = false, empty = true;
    int top = 0;

    push(java.lang.Object):
        requires (!full);
        on_entry {top = top + 1;}
        on_exit {
            empty = false;
            if (top == buffer.length) full = true;
        }

    pop():
        requires (!empty);
        on_entry {top = top - 1;}
        on_exit {
            Full = false;
            if (top == 0) empty = true;
        }
}
DSAL “Bill of Rights”

- Freedom of Expression
  - Syntactic
  - Semantic
- Economic Freedom
  - Cost effective Implementation
  - Cost effective Usage
- Freedom of Assembly
  - DSL Interoperability
- Equality with domain-specific languages and AOP languages
Related Work

- **Language Workbenches**
  - [Fowler, 2005] Language workbenches: The killer-app for domain specific languages.

- **AOP Composition Frameworks**
Conclusion

- A novel design for DSAL workbench that produces first-class DSAL
  - First-class DSL
  - First-class AOP language
- Prototype comprising Spoofax and AWESOME*
- Plug-in for COOL as a first-class DSAL
Thank You!

Arik Hadas and David H. Lorenz
Dept. of Mathematics and Computer Science
The Open University of Israel

arik.hadas@openu.ac.il
https://github.com/OpenUniversity