Language Oriented Modularity: From Theory to Practice

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

Joint Work With:
David H. Lorenz
Language Oriented Modularity (LOM)

LOM is a programming methodology that puts DSALs at the center of the software modularization process.

- **Traditional Process**
  - Modularize your *concerns* with an (aspect) *programming language*

- **The LOM Process**
  - Design (domain specific aspect) *programming languages* for your *concerns*
The Language Oriented Process

- **Works middle-out**
  - Languages are tailored to the problem at hand

- **Similar to LOP**
  - But with DSALs

Language Use
  - Concise

Language Definition
  - Expressive

Language Implementation
  - Reusable
DSALs

- **Domain Specific**
  - Domain-specific abstractions
- **Aspect Oriented**
  - Modularization (weaving) mechanism

**DSLs**

**DSALs**

**AOP**

- COOL
- KALA
- RIDL
- AO4SQL
- AspectGrid
- Racer
In Principle: LOM is similar to LOP

LOP: programming with non-aspect oriented languages

LOM: programming with aspect languages

DSLs

DSALs

AOP

COOL
KALA
RIDL
AspectGrid
AO4SQL
Racer
In Practice: LOM is not Cost-Effective

- **LOM is more costly than LOP**
  - DSALs are more complex to implement
- **LOM is less effective than LOP**
  - The ease of using DSALs (w.r.t. GPALs) is lower than the ease using DSLs (w.r.t. GPLs)

<table>
<thead>
<tr>
<th></th>
<th>DSL</th>
<th>DSAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation cost</td>
<td>🌟</td>
<td>😞</td>
</tr>
<tr>
<td>Effective programming</td>
<td>🌟</td>
<td>😞</td>
</tr>
</tbody>
</table>
In Practice: LOM is not Cost-Effective

- LOM is more costly than that of DSALs
  - DSALs are more complex to implement
- LOM is less effective than that of DSLs
  - The ease of using DSALs (w.r.t. GPLs) is lower than the ease using DSLs (w.r.t. GPLs)

**This Work:** Making LOM Practical

<table>
<thead>
<tr>
<th></th>
<th>DSL</th>
<th>DSAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation cost</td>
<td>😊</td>
<td>😞</td>
</tr>
<tr>
<td>Effective programming</td>
<td>😊</td>
<td>😐</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Problem
- Approach
- Validation
- Evaluation
The Need for DSALs

- Separation of crosscutting concerns
  - Often the right tool for crosscutting concerns found in modern software projects
- Domain-specific abstractions and notations
  - Simpler than general-purpose aspect languages

<table>
<thead>
<tr>
<th></th>
<th>DSL</th>
<th>GPAL</th>
<th>DSAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved Modularity</td>
<td>🎵BMI</td>
<td>🔴BMI</td>
<td>🔴BMI</td>
</tr>
<tr>
<td>Declarative &amp; Simple Syntax</td>
<td>🔴BMI</td>
<td>🎵BMI</td>
<td>🔴BMI</td>
</tr>
</tbody>
</table>
Why *On-Demand* DSALs?

- **DSALs tend to be tightly coupled with the app**
  - Depend on the code structure
  - Depend on the representation of data
- **Less reusable across apps**
  - DSALs are typically application-specific

<table>
<thead>
<tr>
<th>Reusable Across Applications</th>
<th>DSL</th>
<th>DSAL</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Green Smiley" /></td>
<td><img src="image" alt="Green Smiley" /></td>
<td><img src="image" alt="Red Sad" /></td>
</tr>
</tbody>
</table>
DSL Reuse

Application 1

```java
public void logName(Guid id) {
    String name = dao.getName(id);
    logger.log("user " + id + ": " + name);
}
```

Application 2

```java
public void foo(..) {
    // … skipped …
    /* … */ dao.getName(id);
    // … skipped …
}
```

**DSL can be reused across applications**

```
SELECT name FROM users WHERE user_id=<id>;
```
DSL Reuse

Application 1

```java
public void logName(Guid id) {
    String name = dao.getName(id);
    logger.log("user " + id + ": " + name);
}
```

Application 2

```java
public void foo(..) {
    // … skipped …
    /* … */ dao.getName(id);
    // … skipped …
}
```

DSL code

```
SELECT name FROM users WHERE user_id=<id>;
```

DSL can be reused across applications
Aspect in a DSAL called muAudit

```java
logs for com.mucommancher.job.impl.CopyJob:
1. case start log COPY_STARTED with nbFiles baseSourceFolder baseDest
2. case finish log COPY_FINISHED with nbFiles baseSourceFolder baseDest
3. case interrupt log COPY_INTERRUPTED with baseSourceFolder baseDest
4. case pause log COPY_PAUSED with baseSourceFolder baseDestFolder nb
5. case resume log COPY_RESUMED with baseSourceFolder baseDestFolder

logs for com.mucommancher.job.impl.MkdirJob:
6. case start & mkfileMode log MKFile_STARTED with files
7. case start log MKDIR_STARTED with files
8. case finish & mkfileMode log MKFile_FINISHED with files
9. case finish log MKDIR_FINISHED with files
10. case interrupt & mkfileMode log MKFile_INTERRUPTED with files
11. case interrupt log MKDIR_INTERRUPTED with files
12. case pause & mkfileMode log MKFile_PAUSED with files
13. case pause log MKDIR_PAUSED with files
14. case resume & mkfileMode log MKFile_RESUMED with files
15. case resume log MKDIR_RESUMED with files
```
Aspect in a DSAL called muAudit

```java
logs for com.mucommander.job.impl.CopyJob:
2  case start log COPY_STARTED with nbFiles baseSourceFolder baseDest
3  case finish log COPY_FINISHED with nbFiles baseSourceFolder baseDest
4  case interrupt log COPY_INTERRUPTED with baseSourceFolder baseDest
5  case pause log COPY_PAUSED with baseSourceFolder baseDestFolder
6  case resume log COPY_RESUMED with baseSourceFolder baseDestFolder

logs for com.mucommander.job.impl.MkdirJob:
10  case start & mkFileMode log MKFile_STARTED with files
11  case start log MKDIR_STARTED with files
12  case finish & mkFileMode log MKFile_FINISHED with files
13  case finish log MKDIR_FINISHED with files
14  case interrupt & mkFileMode log MKFile_INTERRUPTED with files
15  case interrupt log MKDIR_INTERRUPTED with files
16  case pause & mkFileMode log MKFile_PAUSED with files
17  case pause log MKDIR_PAUSED with files
18  case resume & mkFileMode log MKFile_RESUMED with files
19  case resume log MKDIR_RESUMED with files
```
The Method that Starts File Jobs

```java
/**
 * Starts file job in a separate thread.
 */
public void start() {
    // Return if job has already been started
    if (getState() != FileJobState.NOT_STARTED)
        return;

    // Pause auto-refresh during file job as it potentially may
    // and would potentially cause folder panel to auto-refresh
    getMainFrame().getLeftPanel().getFolderChangeMonitor().set
    getMainFrame().getRightPanel().getFolderChangeMonitor().set

    setState(FileJobState.RUNNING);
    startDate = System.currentTimeMillis();

    jobThread = new Thread(this, getClass().getName());
    jobThread.start();
}
```
Coupling of DSALs with the Base Code

Base Code

```java
public void start() {
    // Return if job has already been started
    if (!started) {
        FileJob.start();
    }
}
```

Aspect Code

```java
case start
    with nbFiles baseSourceFolder baseDestFolder
```
Coupling of DSALs with the Base Code

**Structure Dependency**

**Data Dependency**

**Base Code**

**Aspect Code**

DSALs cannot be easily reused
The Need for LOM

● The need for application specific DSALs calls for LOM
  - Having LOM for DSALs is even more crucial than having LOP for DSLs

● The main obstacle
  - Cost-effectiveness
Outline

- Motivation
- **Problem**
- Approach
- Validation
- Evaluation
DSALs are Second-Class

- Second-class DSLs
  - More costly than ordinary DSLs
- Second-class aspect languages
  - Less effective than ordinary DSLs
- Results in poor cost-effectiveness of LOM
Why DSALs are Second-Class DSLs

- Language workbenches ease DSLs creation
  - Produce a parser for the custom syntax
  - Produce a transformation to some GPL
- But LWs do not help with creation of DSALs
  - Transformations of DSALs typically do not preserve the join-point “fingerprint” (not semantic-preserving)
  - Without a transformation, the weaving semantics are hard to implement
Multi-DSAL Conflicts

- **Foreign Advising**
  - Advise written in one DSAL advises a join point located within an aspect written in a different DSAL

- **Co-Advising**
  - Multiple pieces of advice written in different DSALs advise the same join point within the base code
Why DSALs are 2\textsuperscript{nd}-Class Aspect Langs

- **DSALs lack development tools**
  - Thus complex to edit, browse and compile
- **Hinders effective use of DSALs**
  - Despite being simple and declarative

<table>
<thead>
<tr>
<th></th>
<th>GPALs</th>
<th>DSALs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Editing Tools</td>
<td>😊</td>
<td>😞</td>
</tr>
<tr>
<td>Aspect Development Tools</td>
<td>😊</td>
<td>😞</td>
</tr>
<tr>
<td>Build Tools</td>
<td>😊</td>
<td>😞</td>
</tr>
</tbody>
</table>
# Tool Support for DSALs

<table>
<thead>
<tr>
<th>Language</th>
<th>Support</th>
<th>CF</th>
<th>LW + GPAL</th>
<th>LW + CF</th>
<th>Practical LOM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain-specific</strong></td>
<td><em>DSAL interoperability</em></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td><em>Development process</em></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Aspect-oriented</strong></td>
<td><em>Editing tools</em></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td><em>Aspect development tools</em></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Compilation</em></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CF** = aspect Composition Framework  
**LW** = Language Workbench  
**GPAL** = General Purpose Aspect Language
Outline

- Motivation
- Problem
- Approach
- Validation
- Evaluation
Key Idea

- **Transformation of DSALs into a GPAL annotated with metadata**
  - DSL-like development process for DSALs
  - The use of tools available for the GPAL

![Diagram](image.png)
(1) Metadata for Handling Multi-DSAL Conflicts

- **Hide join points from other DSALs**
  - Forgoing complete obliviousness to prevent foreign advising conflicts

- **Define advice-level ordering values**
  - Fine-grained advice ordering to prevent co-advising conflicts
First-Class Equality with DSLs

- **DSL-like Implementation process for DSALs**
  - Parsing the DSAL code
  - Transforming DSAL code to a GPAL* code
  - No compiler (weaver) modification

- **DSL-like development tools for DSALs**
  - Leverage supportive tools by an existing language workbench
(2) Metadata for Compatibility with GPAL Tools

- Preserve source code location of advice during the transformation
  - To enable browsing and navigation using GPAL tools

- Produce transformations that can be invoked internally by the compiler
  - To enable compilation using GPAL tools
First-Class Equality with Aspect Langs

- Leverage aspect development tools for GPAL
  - Provide browsing and navigation for DSALs
- Leverage compilation tools for the GPAL
  - Enable to build the app the same way with DSALs
- The language workbench produces IDE plugin
  - Providing editing tools for DSALs
Outline

- Motivation
- Problem
- Approach
- Validation
- Evaluation
Extending AspectJ with Metadata

- We extended AspectJ
  - Small set of annotations and interface
- We modified the ajc compiler
  - One-time effort
  - Minimal and optional modifications in case they need to be redone in future releases
- We modified AJDT
  - Minor adjustment
@Hide

Suppress join-points associated with a particular program element

```java
@Target(ElementType.TYPE)
public @interface HideType {
    TypeJoinpoint[] joinpoints() default { TypeJoinpoint.PRE_INIT,
                                           TypeJoinpoint.INIT, TypeJoinpoint.STATIC_INIT,
                                           TypeJoinpoint.WITHIN_INIT, TypeJoinpoint.WITHIN_STATIC_INIT };
}

@Target(ElementType.METHOD)
public @interface HideMethod {
    MethodJoinpoint[] joinpoints() default { MethodJoinpoint.CALL,
                                             MethodJoinpoint.EXECUTION, MethodJoinpoint.WITHIN };
}

@Target(ElementType.FIELD)
public @interface HideField {
    FieldJoinpoint[] joinpoints() default { FieldJoinpoint.SET,
                                             FieldJoinpoint.GET };
}
```
@Order

```java
public @interface Order {
    double value();
}
```

Order advice according to the value of the @Order annotation
@BridgedSourceLocation

Store the original location of advice in the generated code

```java
public @interface BridgedSourceLocation {
    public String file();
    public int line();
    public String module();
}
```

AJDT uses the location pointed to by this annotation, if it exists, as the source location of advise
The compiler uses this interface to identify which files should be transformed (using the extension method) and to transform them internally (convert2java)
Outline

- Motivation
- Problem
- Approach
- Validation
- Evaluation
Evaluation

• Experimental evaluation
  – Implementing new crosscutting feature in the *muCommander* project
  – Separating existing crosscutting concerns in the *oVirt* project

• Comparative evaluation
  – Comparing the implementation of COOL with its implementation in AWESOME
Case Study I: LOM for muCommander

- We applied LOM to muCommander
  - Implemented a new DSAL for an auditing named *muAudit*
  - Implemented aspect solution for auditing of two file operations
About muCommander
Language Definition

Defined in the language grammar definition of Xtext
Language Use

AJDT markers are placed in the base code.

IDE for Eclipse was generated by Xtext, providing editing tools.

AJDT markers are placed in the aspect code.
Language Implementation

```java
@Hide
@BridgedSourceLocation
@HideType
public privileged aspect Logs { 
    @BridgedSourceLocation(line=1, file="/mucommander/src/main/java/ ... /jobs.audit", module="jobs.audit")
    after(CopyJob job): execution(void start()) && this(job) {
        if (true) {
            audit("start copying {0} files from {1} to {2} ({3})", job.nbFiles, job.baseSourceFolder, job.baseDestFolder, job.files);
            return;
        }
    }

    //... skipped ...
}
```
Lessons from Case Study I

• **Implementation in Xtext**
  - Complete implementation in an existing language workbench, like that of a DSL

• **Development tools for programming with muAudit, like those available for a GPAL**
  - Editing tools and aspect dev. tools
  - The project is compiled as if using AspectJ

• **Cost-effective LOM process for a new crosscutting feature (compared to LOP)**
Case Study II: LOM for oVirt

- We implemented DSALs for 3 crosscutting concerns found in the oVirt project
  - Synchronization
  - Permission checks
  - Auditing
About oVirt

Directory Service
- Active Directory
- IPA
- Red Hat Directory Server
- IBM Tivoli Directory Server

Backend
- JBoss by Red Hat

Web Browser Administration Portal
- CLI Shell
- REST API
- Web Browser User Portal

Storage Domain

Web Service
Web App
Web App

VDSM
- libvirt

PostgreSQL

SOAP - Internal Web Service
HTTPS
SSH / SSL

Linux + oVirt Packages

oVirt Node

Console Access

SPICE or VNC
Scattered Code in oVirt

MigrateVmCommand

AddDiskCommand

Synchronization

Auditing

Permissions
Tangled Code in oVirt

CommandBase

```java
private boolean internalCanDoAction() {
    boolean returnValue = false;
    try {
        Transaction transaction = null;
        if (isCanDoActionSupportsTransaction()) {
            transaction = TransactionSupport.suspend();
        }
        try {
            returnValue =
                isUserAuthorizedToRunAction() && isBackwardsCompatible() && validateInputs() && acquireLock()
                && canDoAction() && internalValidateAndSetQuota();
            if (!returnValue && getReturnValue().getCanDoActionMessages().size() > 0) {
                log.warn("CanDoAction of action '{}' failed for user '{}'. Reasons: '{}',
                         getActionType(), getUserName(),
                         StringJoinUtil.join(getReturnValue().getCanDoActionMessages(), ', '));
            }
        } finally {
            if (transaction != null) {
                TransactionSupport.resume(transaction);
            }
        }
    } catch (DataAccessException dataAccessEx) {
        log.error("Data access error during CanDoActionFailure.", dataAccessEx);
        addCanDoActionMessage(EngineMessage.CAN_DO_ACTION_DATABASE_CONNECTION_FAILURE);
    } catch (RuntimeException ex) {
        log.error("Error during CanDoActionFailure.", ex);
        addCanDoActionMessage(EngineMessage.CAN_DO_ACTION_GENERAL_FAILURE);
    } finally {
        if (!returnValue) {
            freeLock();
        }
    }
    return returnValue;
}
```
Example: Using ovirtSync

```java
locks for org.ovirt.engine.core.bll.MigrateVmCommand (async):
    exclusively (overrides) {
        group: VM instance: getVmId
    }
    message: ACTION_TYPE_FAILED_VM_IS_BEING_MIGRATED <"VmName", getVmName>

locks for org.ovirt.engine.core.bll.storage.disk.AddDiskCommand (sync):
    exclusively (overrides) {
        group: VM_DISK_BOOT instance: getVmId if isBooatableDisk
    }
    inclusively (overrides) {
        group: VM instance: getVmId
    }

locks for RunV
```

```java
- RuntimeValuePropertyInfo - com.sun.xml.internal.bind.v2.model.runtime
- RuntimeValuePropertyInfoImpl - com.sun.xml.internal.bind.v2.model.implementations
- RunVmCommand - org.ovirt.engine.core.bll.RunVmCommand
```
Lessons from Case Study II

- **Separated out crosscutting concerns**
  - Scattered code (over 25% of some classes) is encapsulated in a separate module
  - Tangled code is extracted from the root class (over 12% from its LOC)

- **Practical language development**
  - Few hours per DSAL with supportive tools

- **Effective programming with multiple DSALs simultaneously**
  - In a large-scale and complex project
Implementing COOL

• **Language implementation**
  - Implementation of a complex third-party DSAL for thread synchronization

• **Language use**
  - Implementation of an aspect solution for the bounded-stack benchmark example

• **Baseline for comparison**
  - The implementation of COOL in the AWESOME composition framework
Synchronizing a Bounded-Stack

```java
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
    }
}
```

```java
package base;

public class BoundedStackCoord {
    public Coordinator base.BoundedStack {
        selfref {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;
            }
    }
}
```
Avoid Multi-DSAL Conflicts

- **Without @Hide**
  - Known multi-DSAL conflicts reproduced
- **With @Hide**
  - Multi-DSAL conflicts not observed

```java
public aspect AJAuditor {
    pointcut toLog(): call(* *(..)) &&
        !cflow(within(AJAuditor));
    before(): toLog() { log(thisJoinPoint); }
    protected void log(JoinPoint jp) {
        BoundedStack buf = BoundedStack.getInstance();
        try { if (buf != null) buf.add(jp); }
        catch(Exception e) { /* ... skipped ... */ }
}
```
Implementation Effort

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Grammar</th>
<th>Code Transformation</th>
<th>Weaver Plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EV</td>
<td>Other</td>
</tr>
<tr>
<td>Language</td>
<td>SDF</td>
<td>Stratego (AST)</td>
<td>Stratego (AST)</td>
</tr>
<tr>
<td>CF Approach</td>
<td>34</td>
<td>761 (4168)</td>
<td>297 (3001)</td>
</tr>
<tr>
<td>Our Approach</td>
<td>34</td>
<td>0</td>
<td>382 (3008)</td>
</tr>
</tbody>
</table>

- Our implementation vs. the alternative
  - Significantly less code required
  - More high-level
    - AspectJ vs bytecode manipulation
  - Language Workbench compatible
    - Done completely in Spoofax
Related Work

- **Domain Specific Aspect Languages**
  - [Fabry et al., 2015] A Taxonomy of Domain-Specific Aspect Languages.

- **Transformation-based AOP Composition Frameworks**
  - [Shonle et al., 2003] XAspects: An extensible system for domain specific aspect languages.

- **The AWESOME Composition Framework**

- **SpecTackle**
Conclusion

- **LOM can follow a similar process to LOP**
  - For a class of DSALs that are in a sense reducible to a GPAL
- **DSALs become cost-effective**
  - The implementation cost is reduced
  - The effectiveness of using them is increased
- **LOM becomes practical for real-world software development process**

The code is available on GitHub:
https://github.com/OpenUniversity/
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