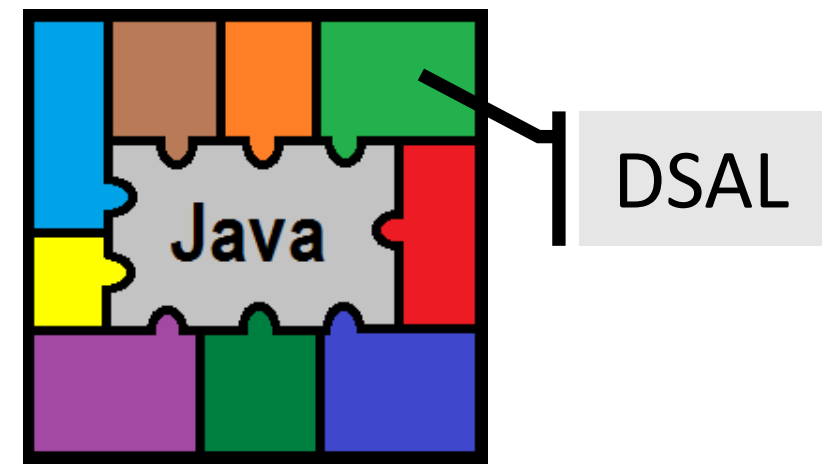


# Experiencing with Language Oriented Modularity

Arik Hadas and David H. Lorenz

Open University of Israel

Language Oriented Modularity (LOM), taking after Language Oriented Programming (LOP), is a programming methodology that involves the development and use of *Domain Specific Aspect Languages (DSALs)* on-demand during the software modularization process. A DSAL is a programming language that is both domain-specific and aspect-oriented. It provides not only domain-specific abstractions and notations like an ordinary *Domain Specific Language (DSL)* does, but also a modularization mechanism for the separation of domain-specific crosscutting concerns.



## Approach: Transforming DSAL code to annotated AspectJ code using a Language Workbench

Hides join points associated with this type to resolve multi-DSAL foreign advising conflicts

Preserves the source location of the advice at the DSAL code to achieve compatibility with AJDT

```
logs for com.mucommander.job.impl.CopyJob:
case start log COPY_STARTED with
nbFiles baseSourceFolder baseDestFolder files
case finish log COPY_FINISHED with
nbFiles baseSourceFolder baseDestFolder
case interrupt log COPY_INTERRUPTED with
baseSourceFolder baseDestFolder
case pause log COPY_PAUSED with
baseSourceFolder baseDestFolder nbProcessedFiles
case resume log COPY_RESUMED with
baseSourceFolder baseDestFolder;
```

```
@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 ...
    }
}
```

## Comparison between our implementation of COOL and the one in the AWESOME composition framework

### Coordinator for a bounded stack

```
coordinator 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;
    }
}
```

### Aspect in AspectJ that uses the stack

```
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 ... */ }
    }
}
```

Resolution of external variables inside the coordinator

Implementation	Grammar	Code Transformation		Weaver Plugin
		EV	Other	
Language	SDF	Stratego (AST)	Stratego (AST)	Java
CF Approach	34	761 (4168)	297 (3001)	1557
Our Approach	34	0	382 (3008)	0

- Significantly less code

- Simpler (High-level programming with AspectJ instead of bytecode manipulation)

- Implementation done completely using the Spoofox language workbench

## Language Oriented Modularity: From Theory to Practice

Research track, Wed, 10:45, D0.07

## Separating crosscutting concerns in the oVirt open source virtualization platform

### Code scattering

```
public class MigrateVmCommand<T extends MigrateVmParameters> ... {
    private VDS destinationVds;
    private EngineError migrationErrorCode;
    private Integer actualDowntime;
    public MigrateVmCommand(T parameters) { ... }
    public MigrateVmCommand(T migrateVmParameters, CommandContext cmdContext) { ... }
    @Override
    protected LockProperties applyLockProperties(LockProperties lockProperties) { ... }
    public final String getDestinationVdsName() { ... }
    public String getDestinationVdsId() { ... }
    protected VDS getDestinationVds() { ... }
    ...
    @Override
    public void runningSucceeded() { ... }
    protected void getDowntime() { ... }
    private void updateVmAfterMigrationToDifferentCluster() { ... }
    private boolean getAutoConverge() { ... }
    private boolean getMigrateCompressed() { ... }
    private int getMaximumMigrationDowntime() { ... }
    private boolean isTunnelMigrationUsed() { ... }
    private boolean isMigrationNetworkUsed() { ... }
    private String getMigrationNetworkAddress(Guid hostId, String migrationNetworkName) { ... }
    protected boolean migrationInterfaceUp(VdsNetworkInterface nic, List<VdsNetworkInterface> nics) { ... }
    @Override
    public AuditLogType getAuditLogTypeValue() { ... }
    private AuditLogType getAuditLogForMigrationStarted() { ... }
    protected AuditLogType getAuditLogForMigrationFailure() { ... }
    protected Guid getDestinationVdsId() { ... }
    protected void setDestinationVdsId(Guid vdsId) { ... }
    @Override
    protected boolean canDoAction() { ... }
    protected void setActionMessageParameters() { ... }
    @Override
    public void rerun() { ... }
    @Override
    protected void reexecuteCommand() { ... }
    protected void determineMigrationFailureForAuditLog() { ... }
    @Override
    protected Guid getCurrentVdsId() { ... }
    public String getDuration() { ... }
    public String getTotalDuration() { ... }
    public String getActualDowntime() { ... }
    @Override
    protected String getLockMessage() { ... }
    private List<Guid> getVdsBlacklist() { ... }
    protected List<Guid> getVdsWhitelist() { ... }
    @Override
    public List<PermissionSubject> getPermissionCheckSubjects() { ... }
    @Override
    public void onPoweringUp() { ... }
}
```

Syntax- errors

Synchronization

AJDT Markers

Auditing

Permissions

Scattered code was separated out (which for some commands exceed 25% of their overall LOC!)

## New crosscutting feature in the muCommander open source file manager

Aspect solution for a missing feature of auditing file operations written in a DSAL

```
*jobs.audit
1 logs for com.mucommander.job.impl.CopyJob:
2 case start log COPY_STARTED with nbFiles baseSourceFolder baseDestFolder files
3 case finish log COPY_FINISHED with nbFiles baseSourceFolder baseDestFolder
4 case interrupt log COPY_INTERRUPTED with baseSourceFolder baseDestFolder nbProcessedFiles
5 case pause log COPY_PAUSED with baseSourceFolder baseDestFolder nbProcessedFiles
6 case resume log COPY_RESUMED with baseSourceFolder baseDestFolder nbProcessedFiles
7 ;
8
9 logs for com.mucommander.job.impl.MkdirJob:
10 case start log MKDIR_STARTED with files
11 case finish log MKDIR_FINISHED with files
12 case interrupt log MKDIR_INTERRUPTED with files
13 case pause log MKDIR_PAUSED with files
14 case resume log MKDIR_RESUMED with files
15 ;
16 ;
17 ;
18 ;
19 ;
20 ;
```

DSAL with easy-to-use syntax for auditing

Auto-Completion

Syntax Highlighting

### Code in the common root of all file operations

```
FileJob.java
200
201 // Return if job has already been started
202 if(getState() != FileJobState.NOT_STARTED)
203     return;
204
205 // Pause auto-refresh during file job as it potentially may
206 // and would potentially cause folder panel to auto-refresh
207 getMainFrame().getLeftPanel().getFolderChangeMonitor().set
208 getMainFrame().getRightPanel().getFolderChangeMonitor().set
209
210 setState(FileJobState.RUNNING);
211 startDate = System.currentTimeMillis();
212
213 jobThread = new Thread(this, getClass().getName());
214 jobThread.start();
215
216
217
218
219
220
221
222
223
224
225
226
```

AJDT Markers

### Code tangling

```
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 && getReturnMessage().size() > 0) {
                log.warn("CanDoAction of action {} failed for user {}. Reasons: {}",
                    getActionType(), getUserName(),
                    StringUtils.join(getReturnMessage().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;
}
```

Tangled code was separated out from the FileJob class (more than 12% of its LOC)



This research was supported in part by the Israel Science Foundation (ISF) under grant No. 1440/14.