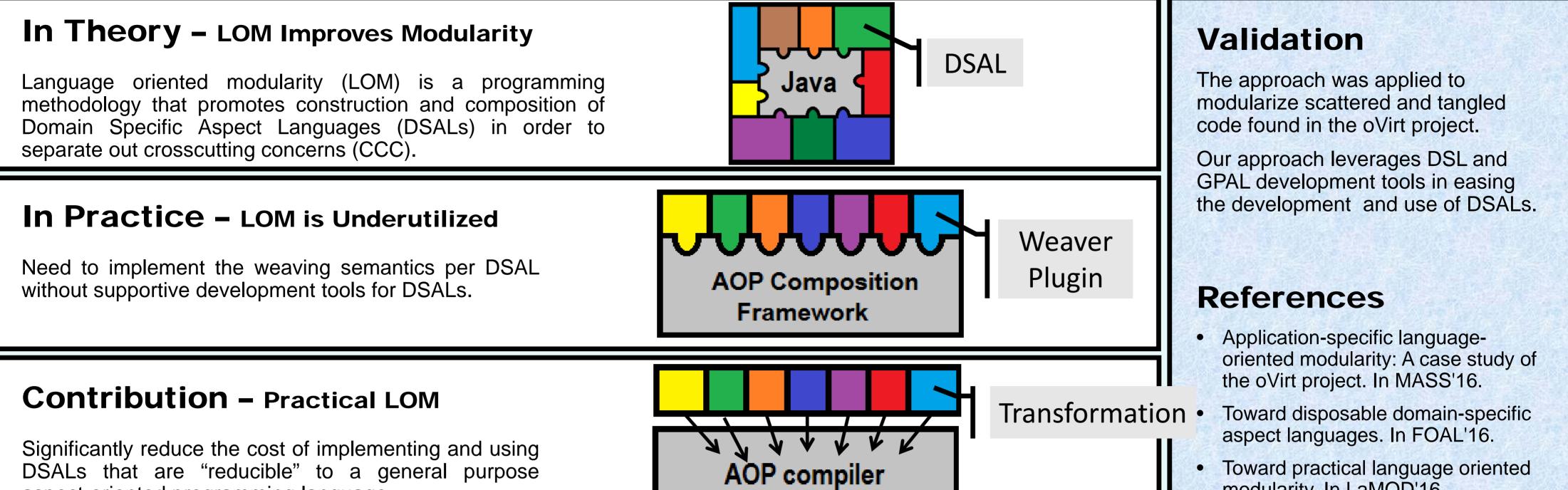# Language Oriented Modularity: From Theory to Practice

Arik Hadas
Open University of Israel

Abstract – Language Oriented Modularity (LOM) is a methodology that is based on Language Oriented Programming (LOP) but applied to Domain-Specific Aspect Languages (DSALs) rather than Domain-Specific Languages (DSLs). LOM involves the construction, composition, and use of DSALs on-demand during the software modularization process, in order to separate out crosscutting concerns (CCC). However, LOM is underutilized and often not used at all in modern projects. The goal of this research is to improve the applicability of LOM for resolving crosscutting concerns in real world projects.

In Theory – LOM Improves Modularity

Language oriented modularity (LOM) is a programming methodology that promotes construction and composition of Domain Specific Aspect Languages (DSALs) in order to separate out crosscutting concerns (CCC).

In Practice – LOM is Underutilized

Need to implement the weaving semantics per DSAL without supportive development tools for DSALs.

Contribution – Practical LOM

Significantly reduce the cost of implementing and using DSALs that are "reducible" to a general purpose aspect-oriented programming language.

DSAL / Java

AOP Composition Framework — Weaver Plugin

AOP compiler — Transformation

Validation

The approach was applied to modularize scattered and tangled code found in the oVirt project.

Our approach leverages DSL and GPAL development tools in easing the development and use of DSALs.

References

- Application-specific language-oriented modularity: A case study of the oVirt project. In MASS'16.
- Toward disposable domain-specific aspect languages. In FOAL'16.
- Toward practical language oriented modularity. In LaMOD'16.

Xcutting Problem

Synchronization

Auditing

Permissions

AJDT Markers

Syntax-errors

Auto-Completion

LOM Solution

DSAL with easy-to-use syntax for auditing

Transforming DSAL code to a GPAL aspect

Implementing code transformation using a Language Workbench

Generate IDE Support for programming with the DSAL

DSAL with easy-to-use syntax for synchronization

Generated aspect for auditing

Syntax Highlighting

	LOP	LOM	Practical LOM
Languages	DSLs	DSALs	DSALs reducible to a GPAL
Compiler	Host compiler	Per-DSAL weaver	Kernel weaver
IDE Tools for Language Dev.	Language Workbench	None for weaving semantics	Language Workbench
IDE Tools for Language Use	Language Workbench	Manual	Language Workbench
Aspect Dev. Tools	N/A	Per-DSAL	AJDT

Approach

Extend AspectJ to serve as a kernel language for LOM

```
public privileged aspect Logs {
    @BridgedSourceLocation(line=1,
      file="/bll/src/main/java/org/ovirt/engine/core/bll/ovirt.audit",
      module="ovirt.audit")
    AuditLogType around(org.ovirt.engine.core.bll.MigrateVmCommand command):
        execution(* getAuditLogTypeValue()) && this(command) {
        if (command.getSucceeded() && command.isReturnValueUp())
          return AuditLogType.VM_MIGRATION_DONE;
        if (command.getSucceeded() && command.isInternalExecution())
          return AuditLogType.VM_MIGRATION_START_SYSTEM_INITIATED;
        if (command.getSucceeded())
          return AuditLogType.VM_MIGRATION_START;
        if (!command.getSucceeded() && command.isHostInPrepareForMaintenance())
          return AuditLogType.VM_MIGRATION_FAILED_DURING_MOVE_TO_MAINTENANCE;
        if (!command.getSucceeded())
          return AuditLogType.VM_MIGRATION_FAILED;
        return AuditLogType.UNASSIGNED;
    }

    @BridgedSourceLocation(line=14,
      file="/bll/src/main/java/org/ovirt/engine/core/bll/ovirt.audit",
      module="ovirt.audit")
    AuditLogType around(org.ovirt.engine.core.bll.storage.export.ExportVmTemplateCommand
    command):
        execution(* getAuditLogTypeValue()) && this(command) {
        if (command.getSucceeded() && command.getActionState() == CommandActionState.EXECUTE)
          return AuditLogType.IMPORTEXPORT_STARTING_EXPORT_TEMPLATE;
        if (!command.getSucceeded() && command.getActionState() == CommandActionState.EXECUTE)
          return AuditLogType.IMPORTEXPORT_EXPORT_TEMPLATE_FAILED;
        if (command.getSucceeded()
            && command.getActionState() == CommandActionState.END_SUCCESS)
          return AuditLogType.IMPORTEXPORT_EXPORT_TEMPLATE;
        if (!command.getSucceeded()
            && command.getActionState() == CommandActionState.END_SUCCESS)
          return AuditLogType.IMPORTEXPORT_EXPORT_TEMPLATE_FAILED;
        return proceed(command);
    }

    @BridgedSourceLocation(line=25,
      file="/bll/src/main/java/org/ovirt/engine/core/bll/ovirt.audit",
      module="ovirt.audit")
    AuditLogType around(org.ovirt.engine.core.bll.storage.disk.AddDiskCommand command):
        execution(* getAuditLogTypeValue()) && this(command) {
        ... skipped ...
    }
}
```

STUDENT RESEARCH COMPETITION

Association for Computing Machinery

sponsored by Microsoft Research