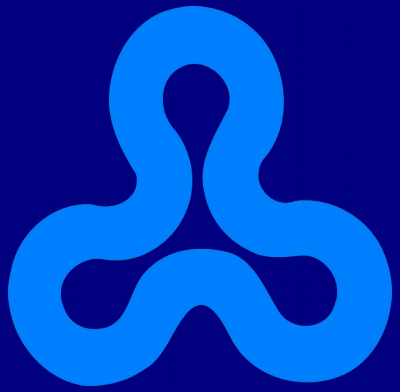
Language Oriented Modularity: From Theory to Practice

Arik Hadas

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



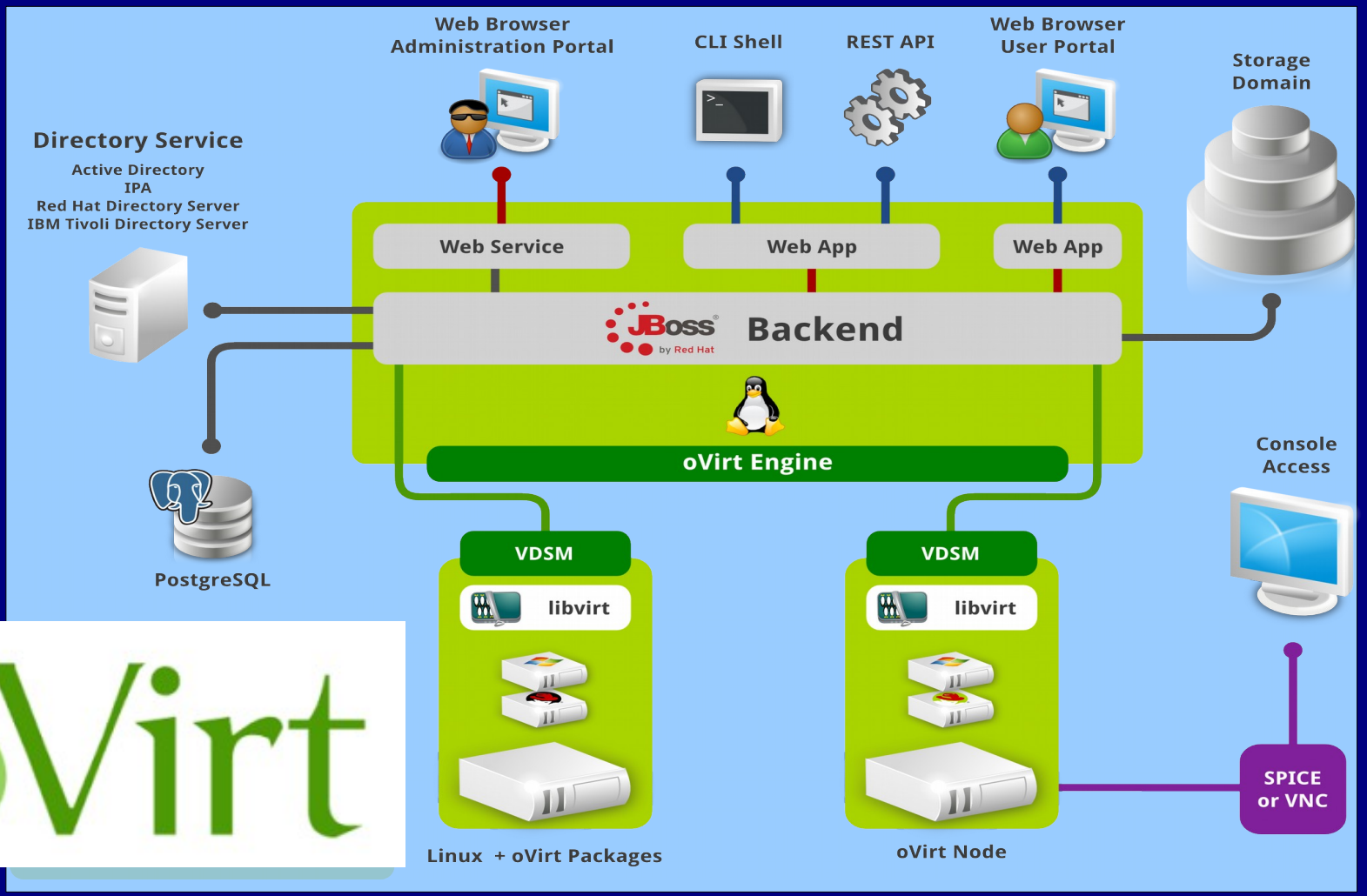
Adviser:
David H. Lorenz

Language Oriented Modularity (LOM)

- **Methodology**
  - For modularization via development and use of Domain Specific Aspect Languages (DSALs)
- **Theory**
  - Very powerful methodology
- **Practice**
  - Underutilized in modern projects

oVirt – Open Virtualization

- Open source enterprise application for providing and managing virtual data centers



oVirt

Scattered Code in oVirt

MigrateVmCommand

```
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 getDueToMigrationError() { ... }
    protected VDS getDestinationVds() { ... }
    @Override
    protected void processVmOnDown() { ... }
    protected boolean initVdss() { ... }
    private List<Guid> getDestinationHostList() { ... }
    @Override
    protected void executeVmCommand() { ... }
    private boolean perform() { ... }
    private boolean migrateVm() { ... }
    private MigrateVDSCommandParameters createMigrateVDSCommandParameters() { ... }
    @Override
    public void runningSucceded() { ... }
    protected void getDowntime() { ... }
    private void updateVmAfterMigrationToDifferentCluster() { ... }
    private Boolean getAutoConverge() { ... }
    private Boolean getMigrateCompressed() { ... }
    private int getMaximumMigrationDowntime() { ... }
    private boolean isTunnelMigrationUsed() { ... }
    private String getMigrationNetworkIp() { ... }
    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 onPowerringUp() { ... }
}
```

synchronization

Auditing

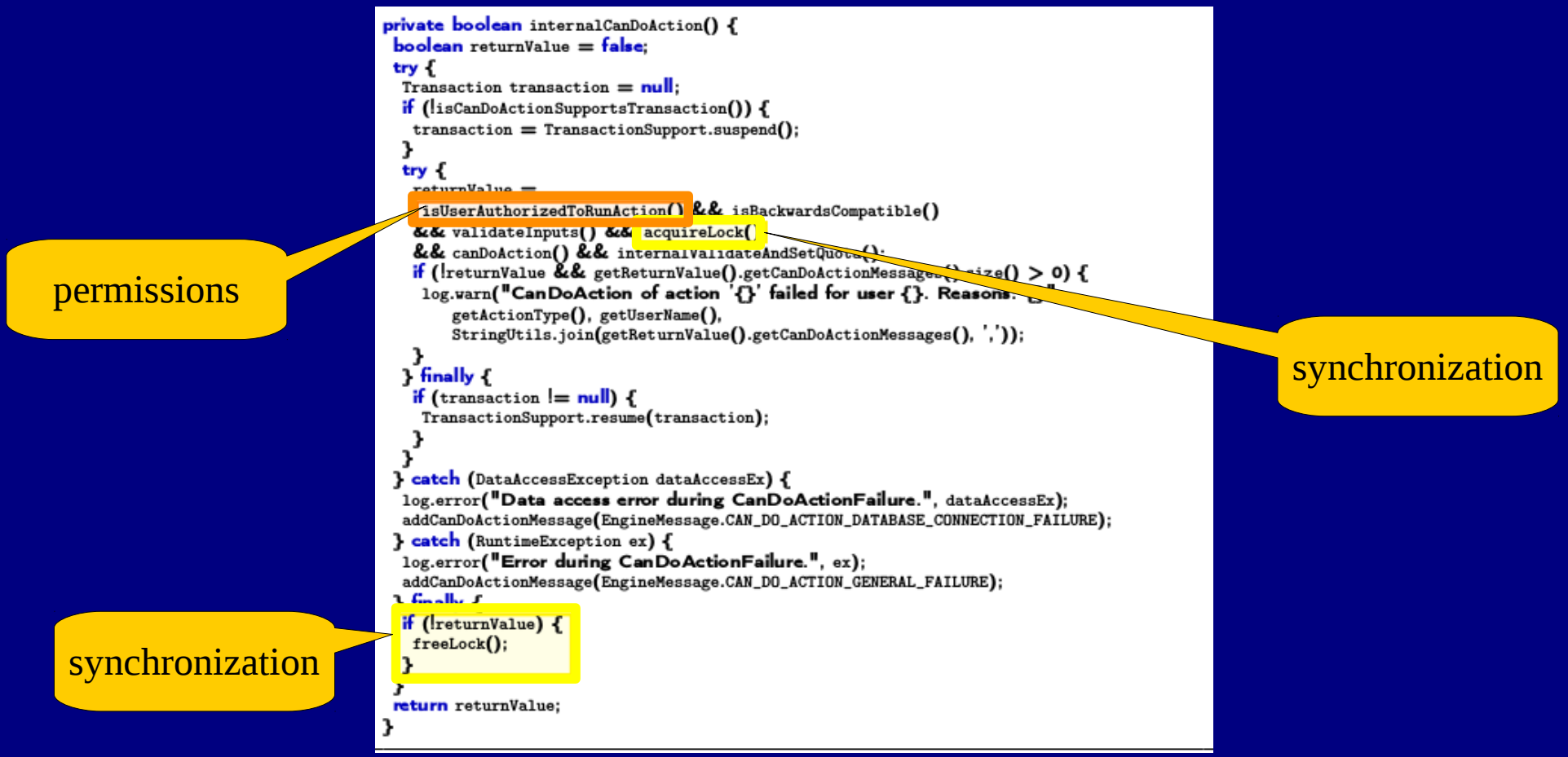
Permissions

AddDiskCommand

```
public class AddDiskCommand<T extends AddDiskParameters> ... {
 protected AddDiskCommand(Guid commandId) { ... }
 public AddDiskCommand(T parameters) { ... }
 public AddDiskCommand(T parameters, CommandContext commandContext) { ... }
 @Override
 protected boolean canDoAction() { ... }
 protected boolean checkIfLunDiskCanBeAdded(DiskValidator diskValidator) { ... }
 protected boolean checkIfImageDiskCanBeAdded(VM vm, DiskValidator diskValidator) { ... }
 private boolean isShareableDiskOnGlusterDomain() { ... }
 private boolean canAddShareableDisk() { ... }
 private boolean checkExceedingMaxBlockDiskSize() { ... }
 private boolean isStoragePoolMatching(VM vm) { ... }
 protected boolean checkImageConfiguration() { ... }
 private double getRequestDiskSpace() { ... }
 @Override
 protected boolean isVmExist() { ... }
 private DiskImage getDiskImageInfo() { ... }
 private boolean isExceedMaxBlockDiskSize() { ... }
 protected DiskLunMapDao getDiskLunMapDao() { ... }
 protected DiskImageDynamicDao getDiskImageDynamicDao() { ... }
 private Guid getDisksStorageDomainId() { ... }
 @Override
 public Guid getStorageDomainId() { ... }
 @Override
 public List<PermissionSubject> getPermissionCheckSubjects() { ... }
 protected void setActionMessageParameters() { ... }
 @Override
 protected void executeVmCommand() { ... }
 private void createDiskBasedOnLun() { ... }
 protected VmDevice addManagedDeviceForDisk(Guid diskId, Boolean isUsingScsiReservation) {
     ... }
 protected VmDevice addManagedDeviceForDisk(Guid diskId) { ... }
 protected boolean shouldDiskBePlugged() { ... }
 private void createDiskBasedOnImage() { ... }
 private void createDiskBasedOnCinder() { ... }
 private VdcActionParametersBase buildAddCinderDiskParameters() { ... }
 private void setVmSnapshotIdForDisk(AddImageFromScratchParameters parameters) { ... }
 private void addDiskPermissions(Disk disk) { ... }
 @Override
 public AuditLogType getAuditLogTypeValue() { ... }
 private boolean isDiskStorageTypeRequiresExecuteState() { ... }
 private AuditLogType getExecuteAuditLogTypeValue(boolean successful) { ... }
 protected AuditLogType getEndSuccessAuditLogTypeValue(boolean successful) { ... }
 @Override
 protected VdcActionType getChildActionType() { ... }
 @Override
 protected List<Class<?>> getValidationGroups() { ... }
 @Override
 protected Map<String, Pair<String, String>> getSharedLocks() { ... }
 @Override
 protected Map<String, Pair<String, String>> getExclusiveLocks() { ... }
 protected void setLoggingForCommand() { ... }
 private Guid getQuotaId() { ... }
 @Override
 protected void endSuccessfully() { ... }
 private void plugDiskToVmIfNeeded() { ... }
 protected boolean setAndValidateDiskProfiles() { ... }
 @Override
 public List<QuotaConsumptionParameter> getQuotaStorageConsumptionParameters() { ... }
 protected StorageDomainValidator createStorageDomainValidator() { ... }
}
```

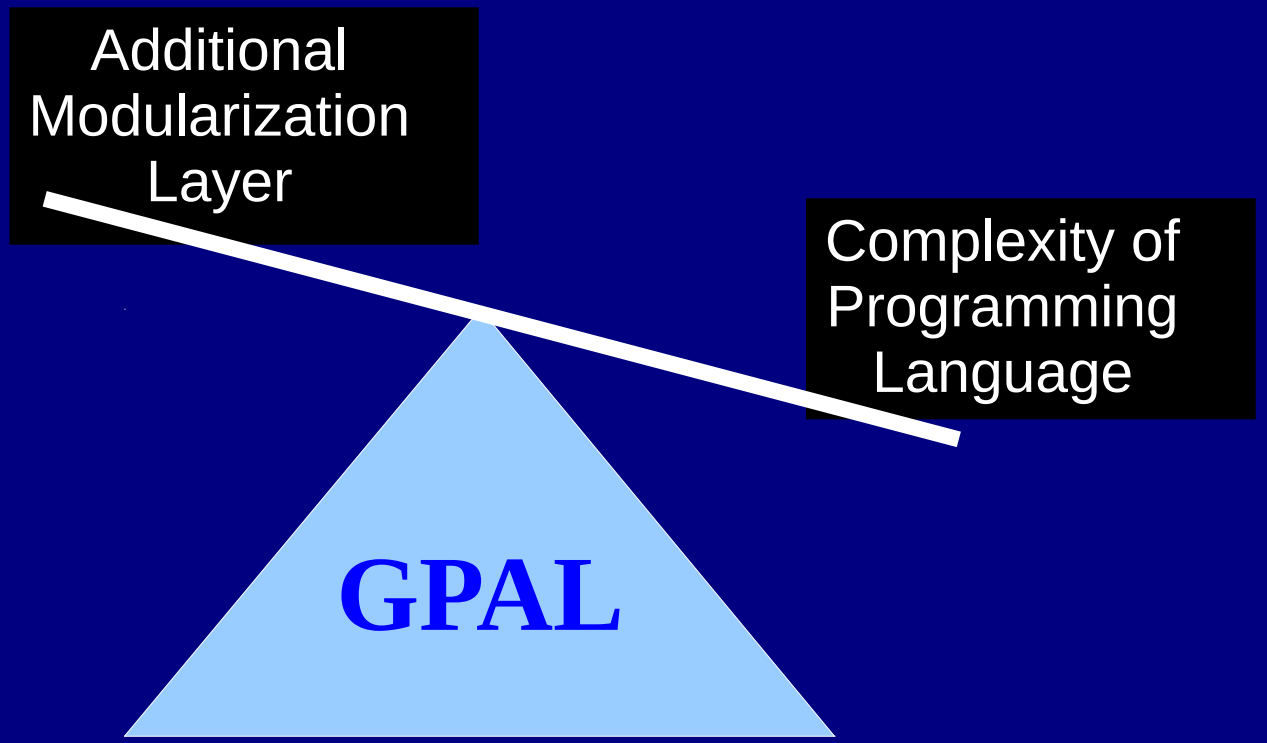
Tangled Code in oVirt

CommandBase



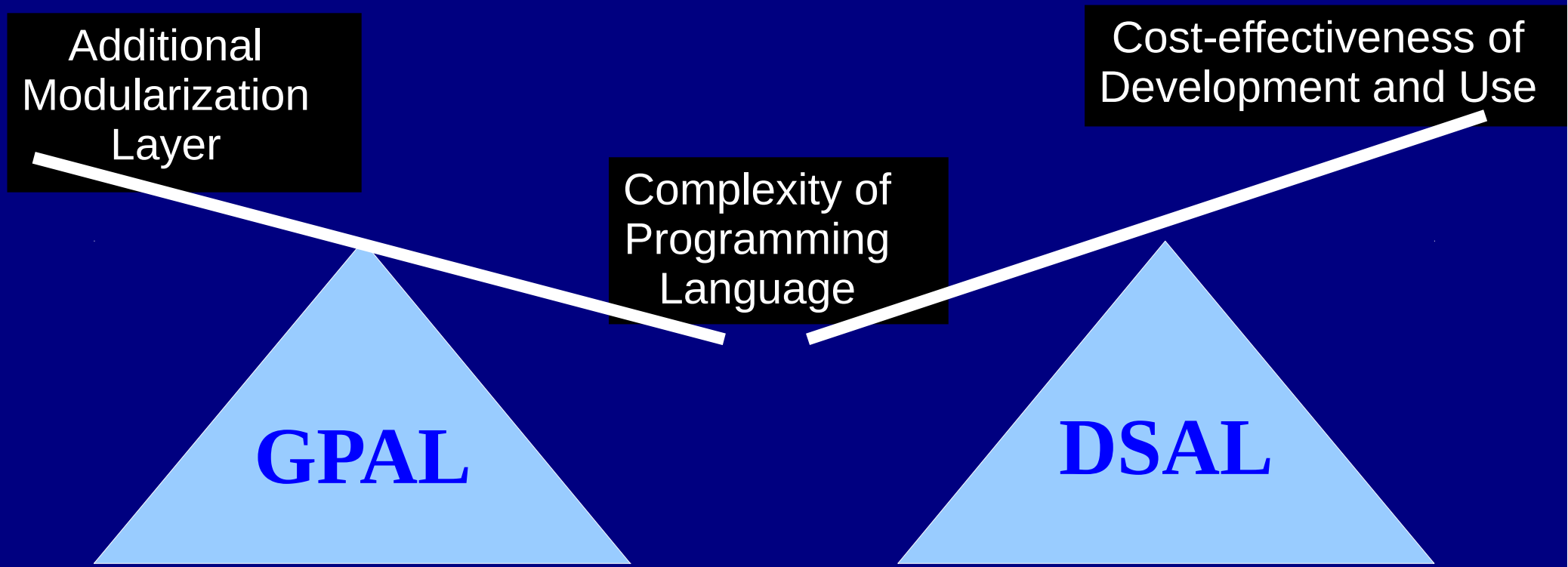
Problem in a Nutshell

- **General Purpose Aspect Languages (GPALs)**
  - **Too complex to use**



Problem in a Nutshell

- **General Purpose Aspect Languages (GPALs)**
  - **Too complex to use**
- **Domain Specific Aspect Languages (DSALs)**
  - **Too complex to develop**



Contribution in a Nutshell

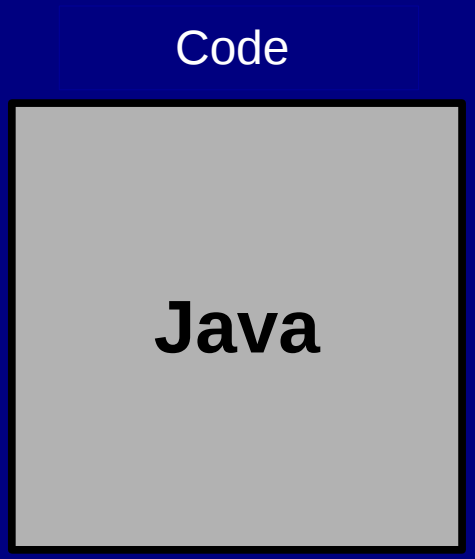
- **Practical LOM**
  - **Make the DSAL development process more like that of DSLs**

Outline

- Introduction
- **Problem**
- Approach
- Evaluation
- Conclusion

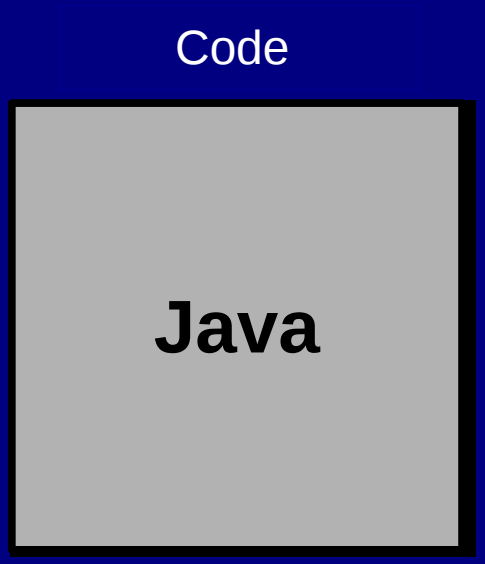
Language Oriented Modularity (LOM)

- **A methodology that puts Domain Specific Aspect Languages (DSALs) at the center of the software modularization process.**



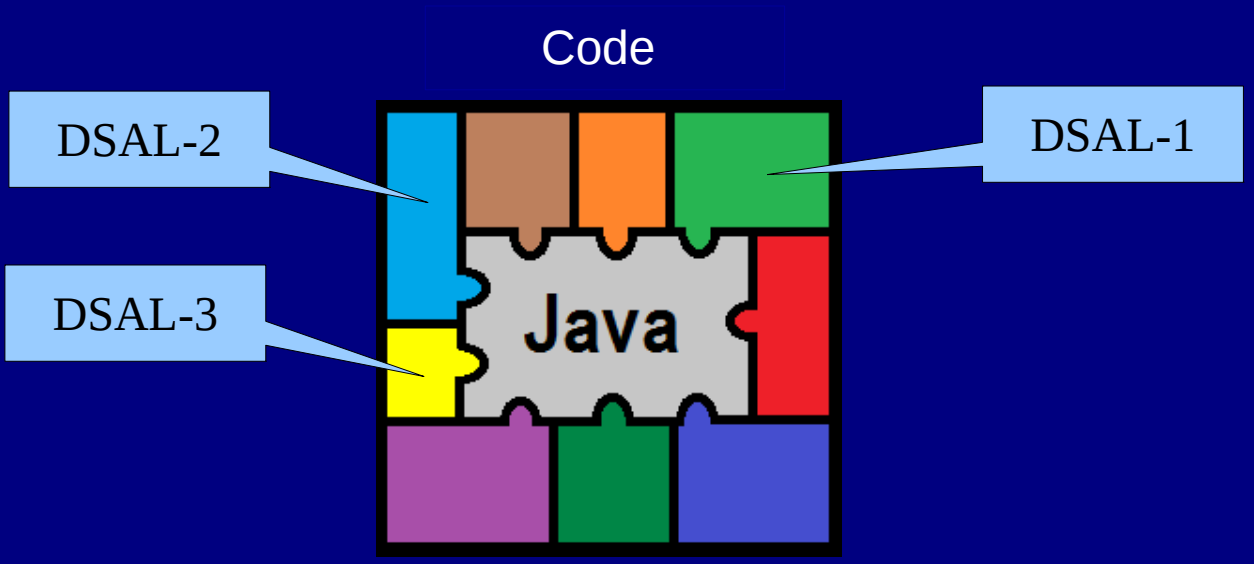
Language Oriented Modularity (LOM)

- **A methodology that puts Domain Specific Aspect Languages (DSALs) at the center of the software modularization process.**
  - **On-demand development and use of DSALs**



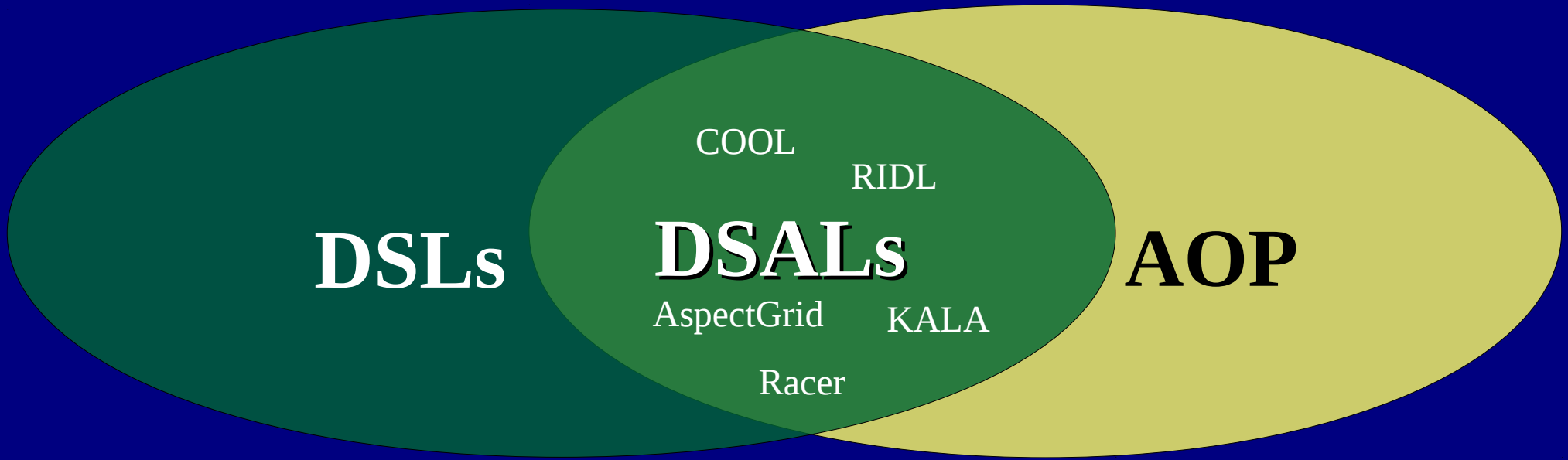
Language Oriented Modularity (LOM)

- **A methodology that puts Domain Specific Aspect Languages (DSALs) at the center of the software modularization process.**
  - **On-demand development and use of DSALs**



Pros of LOM

- **Separation of crosscutting concerns**
  - **Improved software modularity compared to GPLs or DSLs**
- **Domain specific languages**
  - **Programming in more declarative and simpler languages than GPALs**



Cons of LOM

- **Cost**
  - **Definition and implementation cost is higher**
- **Effectiveness**
  - **Use of DSALs (compared to GPALs) is less effective than DSLs (compared to GPLs)**

	DSLs	DSALs
Cost-effectiveness		

Working Hypothesis

- **Making LOM more like LOP could make LOM more practical**
  - **DSALs more like DSLs (definition; implementation)**
  - **DSALs more like GPALs (use)**

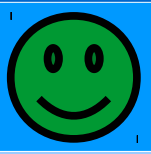
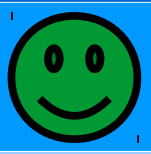
	DSLs	DSALs	GPAL
Definition; Implementation			
Use			

Problem Preview

	DSLs	DSALs
Language Definition		
Language Implementation		
Language Use		

Language Definition

- **Syntax**
  - **Domain-specific notations and abstraction**
- **Semantics**
  - **Complex to define the weaving semantics when multiple DSALs are being used simultaneously**

	DSLs	DSALs
Domain-Specific Syntax		
Weaving Semantics	Not Needed	

Language Implementation

- **Language workbenches are for DSLs**
  - Produces a parser for the custom syntax
  - Produces a transformation to some GPL
- **No equivalent tool for DSALs**
  - The implementation of weaving semantics is generally a costly task

	DSLs	DSALs
Parsing		
Compilation		

Language Use

- **Programming with a DSL**
  - Language workbench produces editing tools
- **Programming with a DSAL**
  - Simpler language but lacks development tools

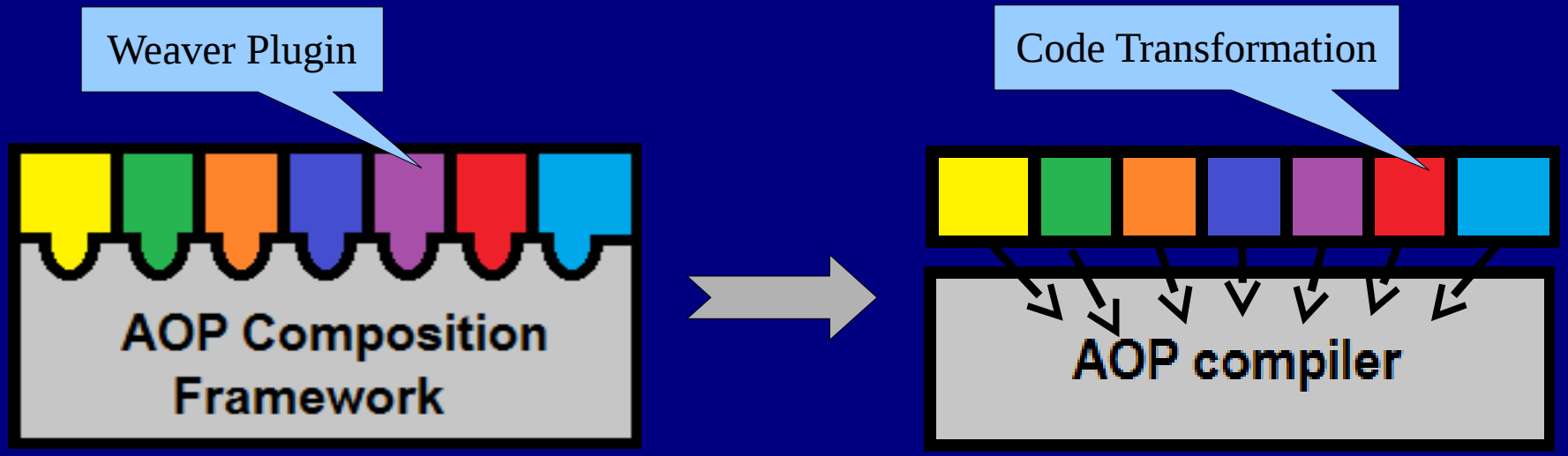
	DSLs	DSALs
Common Editing Tools		
Build Tools		
Aspect Development Tools	Not Needed	

Outline

- Introduction
- Problem
- **Approach**
- Evaluation
- Conclusion

Key Idea

- **Transform DSALs into a kernel language that is based on a GPAL**
  - **No need to implement a weaver per DSAL**
  - **Aspect development tools for the GPAL would work with the DSAL code**



Transformation-based Approach

- **Restriction on crosscutting concerns**
  - CCC that could be modularized using a GPAL
- **DSALs can be transformed into that GPAL**
  - Aspect development tools for the GPAL would work with the DSAL code
  - Most of the developers program with simpler and more declarative languages

GPAL-based Kernel Language

- The kernel language provides constructs for resolving possible multi-DSALs conflicts
  - Hide joinpoint shadows in order to resolve *foreign advising* issues
  - Sort advise to resolve *co-advising* issues
- During transformation of DSAL code these constructs can be defined declaratively
  - Annotate join points that should be hidden
  - Annotate advice so they could be sorted
- The simpler the DSALs are, the less common these conflicts are

Leveraging Language Workbench

- Most of the DSAL development can be done using a language workbench
  - Grammar definition for the DSAL
  - Transformation of the DSAL to the kernel language
- Supportive tools provided by a language workbench
  - Reduce the implementation cost
- Editing tools for programming with the DSALs
  - Generated by the language workbench

Outline

- Introduction
- Problem
- Approach
- **Evaluation**
- Conclusion

LOM for oVirt

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

Demonstration - oVirtSync

- Developing a DSAL for synchronization in oVirt:

  https://youtu.be/uj80yWutQak
- Resolving synchronization in oVirt with DSAL:

  https://youtu.be/PTy9rYDQSo4
- The code is available on GitHub

  https://github.com/OpenUniversity

Implementation Effort

- One time effort
  - Compiler for the kernel language
- Per-application effort
  - Compile oVirt with AspectJ compiler
- The produced DSALs were
  - Relatively easy to define
  - Relatively easy to implement
  - Relatively easy to use

Outline

- Introduction
- Problem
- Approach
- Evaluation
- Conclusion

Related Work

- **Domain Specific Aspect Languages**
  - [Fabry at al., 2015] A Taxonomy of Domain-Specific Aspect Languages.
- **Transformation-based AOP Composition Frameworks**
  - [Shonle at al., 2003] XAspects: An extensible system for domain specific aspect languages.
  - [Tanter, 2006] Aspects of composition in the Reflex AOP kernel.
- **SpecTackle**
  - [Lorenz and Mishali, 2012] SpecTackle: Toward a specification based DSAL composition process.

Summary

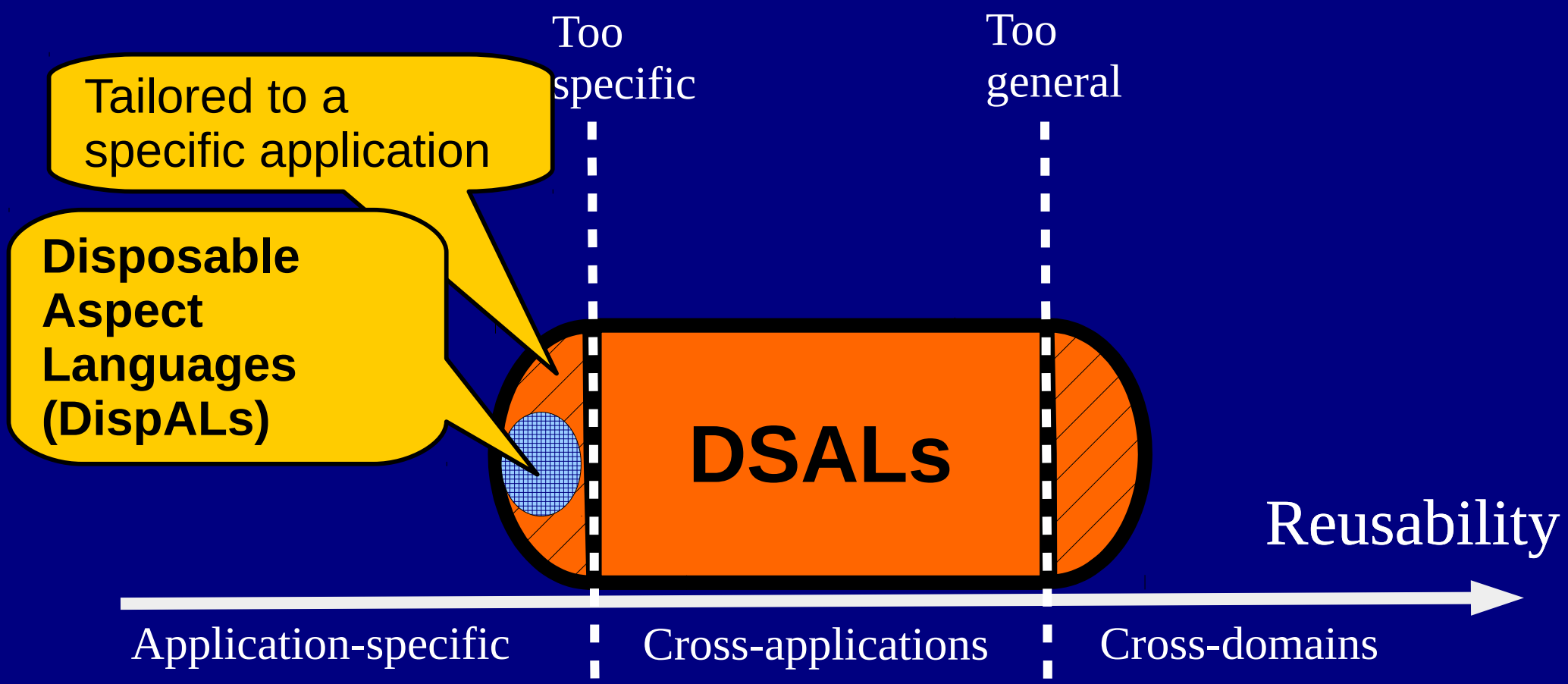
- We bring the DSAL development process one step closer to the development process of DSLs
  - For a class of DSALs that are in a sense reducible to a GPAL
- That way, their cost-effectiveness is improved
  - The implementation cost is reduced
  - The definition cost could be reduced
  - The effectiveness of using them is increased
- That may make the LOM methodology practical for real-world software development process

Conclusion

- New classes of DSALs
  - Application specific
  - Disposable

Conclusion

- **New classes of DSALs**
  - **Application specific**
  - **Disposable**



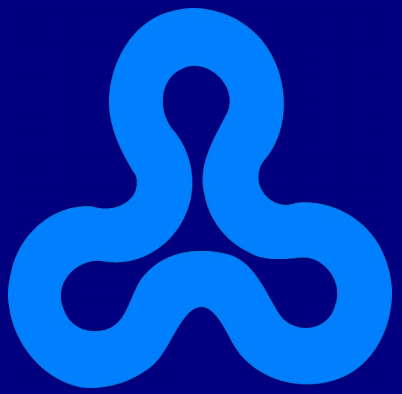
Conclusion

- New classes of DSALs
  - Application specific
  - Disposable
- Challenge general conception of language design
  - Lower reuse may improve cost-effectiveness

Conclusion

- New classes of DSALs
  - Application specific
  - Disposable
- Challenge general conception of language design
  - Lower reuse may improve cost-effectiveness
- Agile-like software modularization process
  - Start with disposable DSALs and gradually move to reusable DSALs

Thank You!



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

arik.hadas@openu.ac.il

https://github.com/OpenUniversity