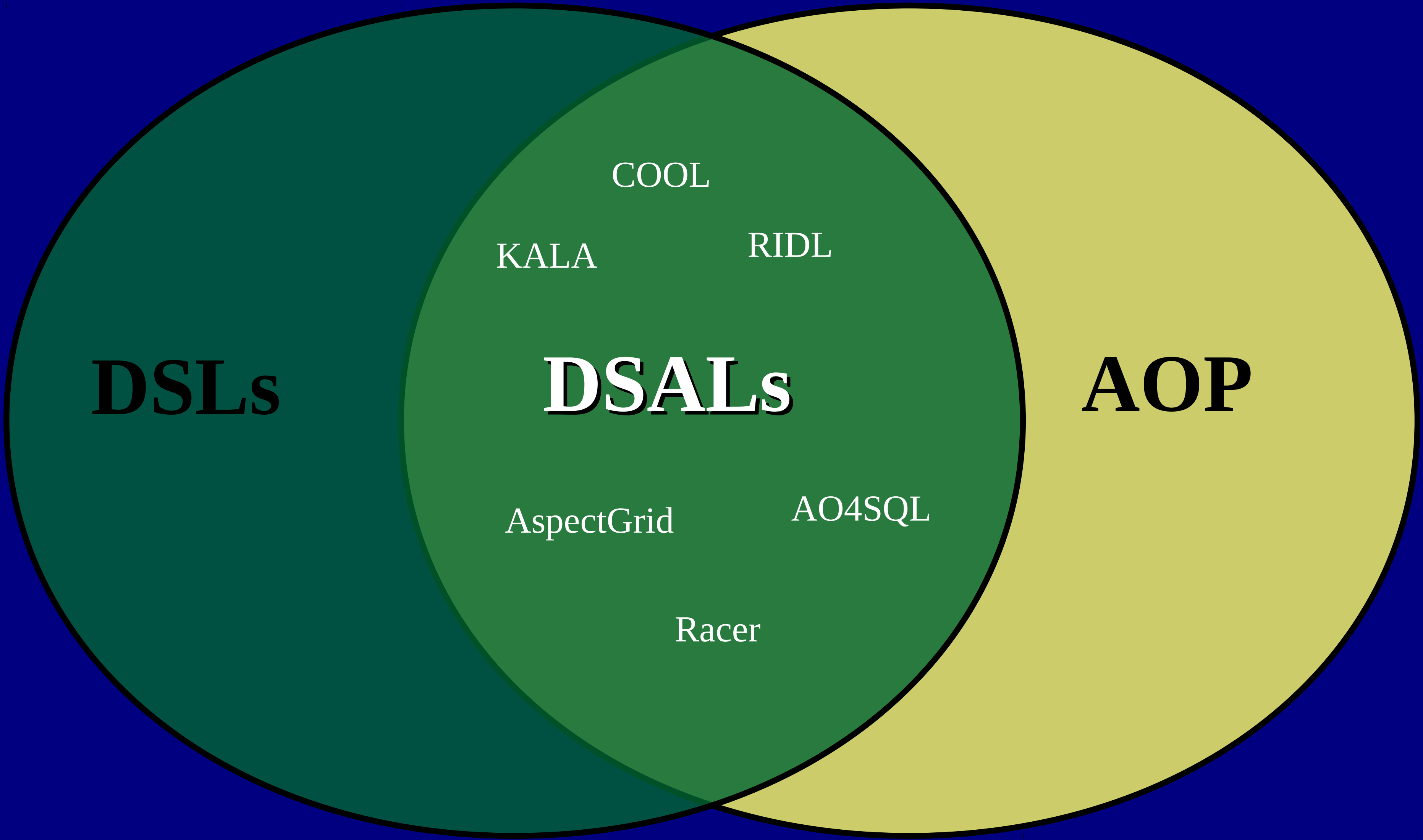
Toward Practical Language Oriented Modularity

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

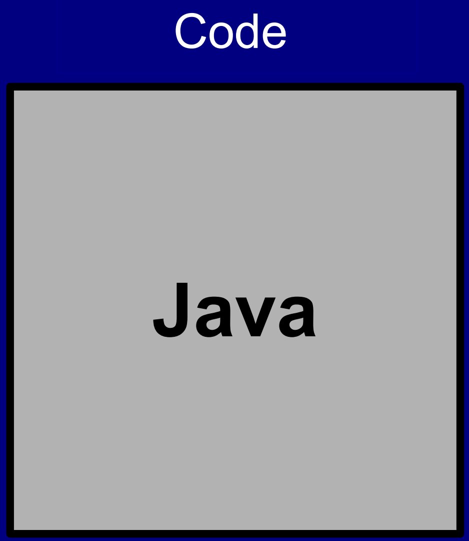
Joint Work With:
David H. Lorenz

Domain Specific Aspect Languages



Language Oriented Modularity (LOM)

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

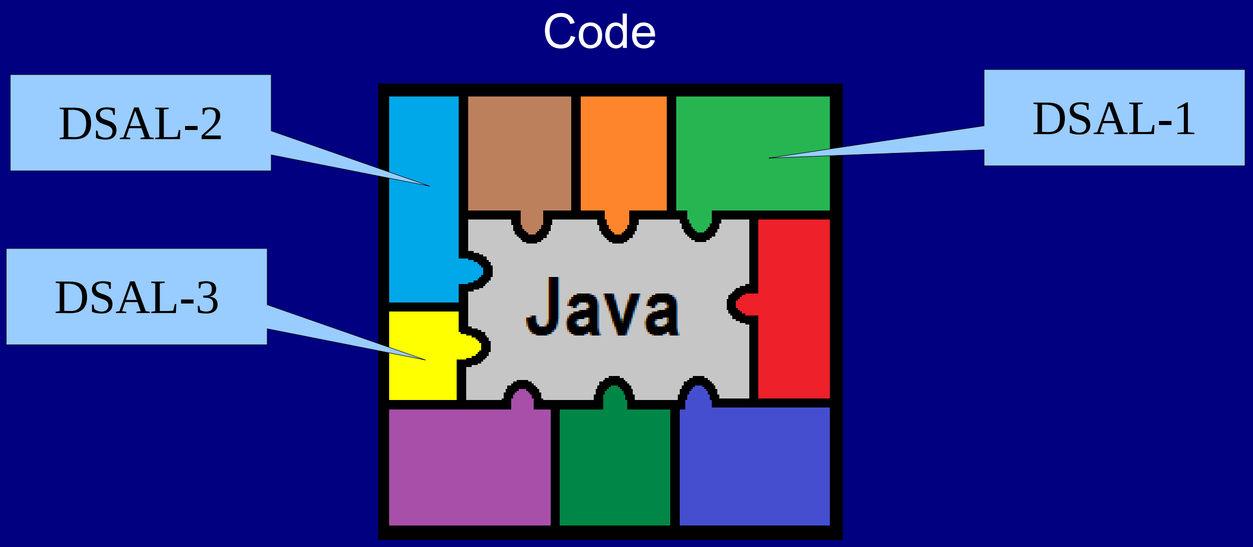


Code

Java

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

- Domain specific languages
  - Programming with more declarative and simpler languages than general purpose aspect languages (GPALs)
- Separation of crosscutting concerns
  - Improved software modularity compared to general purpose languages or DSLs

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)

	LOP & DSLs	LOM & DSALs
Cost-effectiveness	🙂	☹

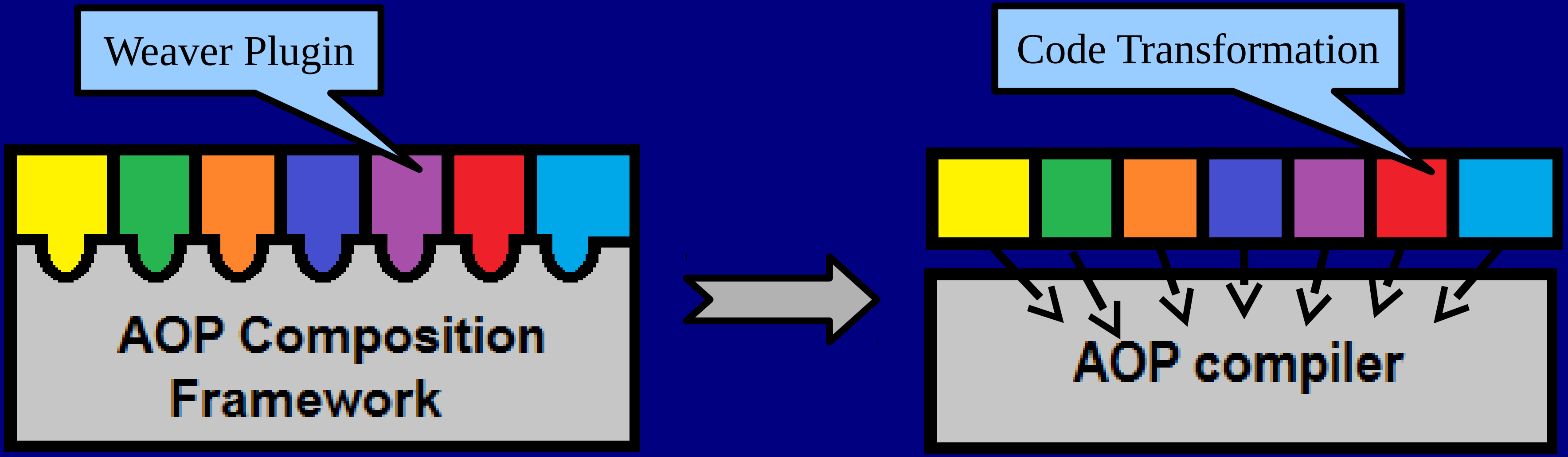
Working Hypothesis

- 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		☹	☺

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



Outline

- Introduction
- Problem
- Solution
- Evaluation
- Conclusion

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
- Solution
- Evaluation
- Conclusion

Solution Preview

	LOM & DSALs	Practical LOM
Language Definition	😐	😐
Language Implementation	☹	☺
Language Use	☹	☺

Transformation-based Approach

- Restriction on crosscutting concerns
  - CCC that could be modularized using a GPAL
- Transform DSALs into a kernel language that is based on a GPAL
  - DSALs can be transformed into that GPAL
  - No need to implement a weaver per DSAL
  - Aspect development tools for the GPAL would work with the DSAL code

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
- The supportive tools provided by a language workbench reduce the implementation cost
- Editing tools for programming with the DSALs can be generated by the language workbench

Outline

- Introduction
- Problem
- Solution
- Evaluation
- Conclusion

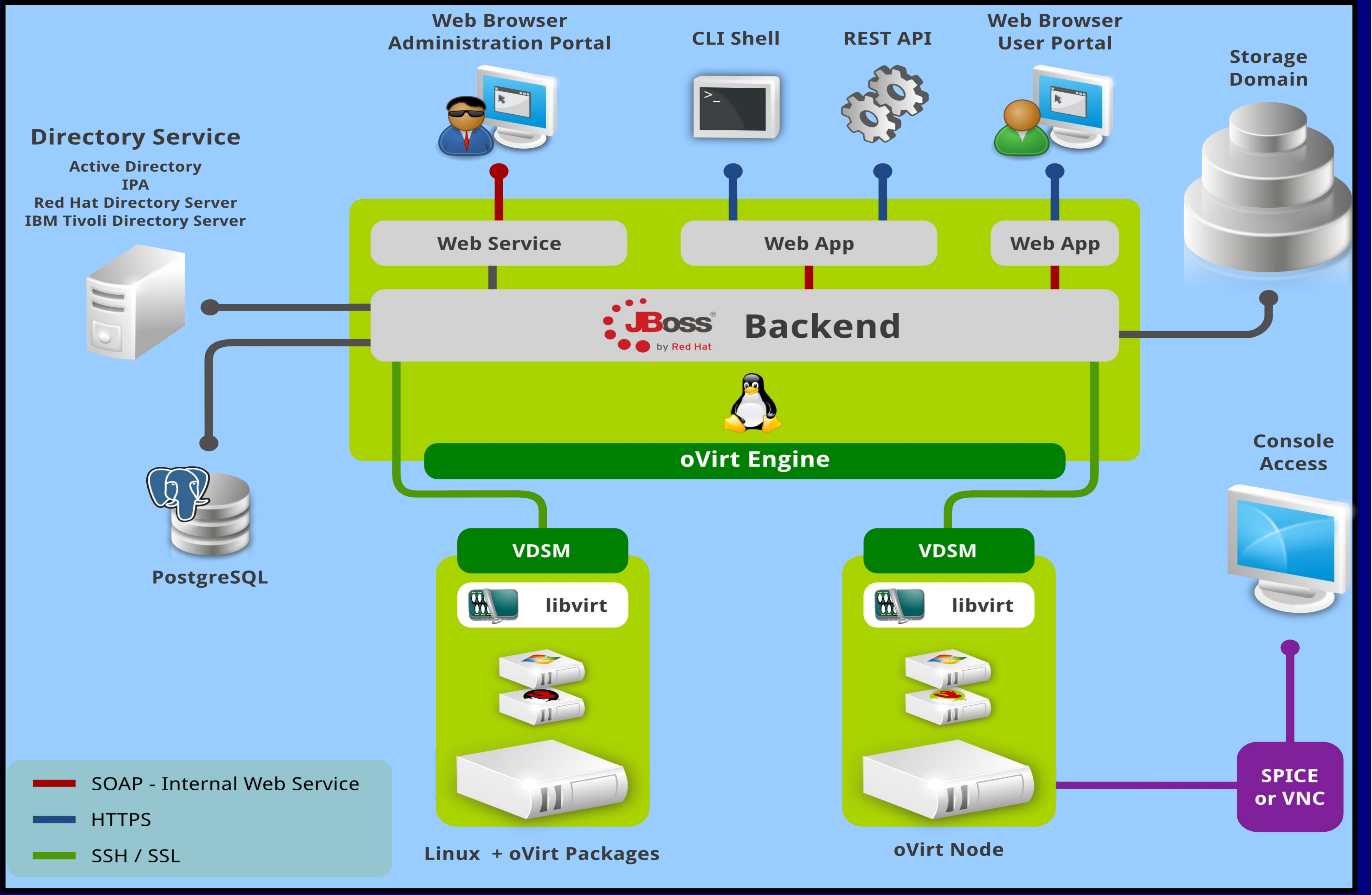
Practical LOM in oVirt

- We applied LOM to oVirt
  - Implemented a DSAL named oVirtSync
  - Used oVirtSync to modularize synchronization in the oVirt project
- Experience
  - Relatively easy to define
  - Relatively easy to implement
  - Relatively easy to use

oVirt – Open Virtualization

- oVirt
  - Open-source enterprise application for providing and managing virtual data centers
  - The upstream of Red Hat Enterprise Virtualization
  - Alternative to VMware's vSphere
- oVirt-Engine
  - The control center of oVirt
  - Executes operations it gets from clients
  - Reports the up-to-date status of the data center

oVirt-Engine in oVirt's Architecture



🟥 SOAP - Internal Web Service

🟦 HTTPS

🟩 SSH / SSL

Synchronization in oVirt-Engine

- The core design of oVirt-Engine is based on the COMMAND design pattern
  - All commands inherit from a common root class
  - Synchronous and asynchronous commands
- Some commands cannot be executed simultaneously
  - oVirt-Engine prevents such conflicts
  - Special locking mechanism was implemented
  - Such conflict produced an error message that is returned to the client

Crosscutting Concern Problem

- We have found that synchronization related code crosscut many modules in oVirt-Engine
  - Scattered across most of the commands
    - Defines the entities to lock, scope of the locks, error messages, etc.
  - Tangled in the common root, CommandBase
    - When to acquire locks, how to build locks, when to release locks, etc.

Demonstration – oVirtSync

https://youtu.be/uj80yWutQak

Outline

- Introduction
- Problem
- Solution
- Evaluation
- Conclusion

Related Work

- DSAL Workbench
  - [Hadas and Lorenz, 2015] Demanding first-class equality for 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

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