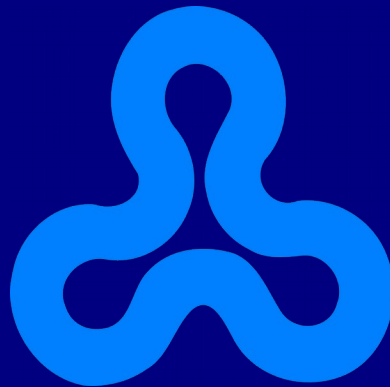


# A Language Workbench for Implementing Your Favorite Extension to AspectJ

**Arik Hadas**

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

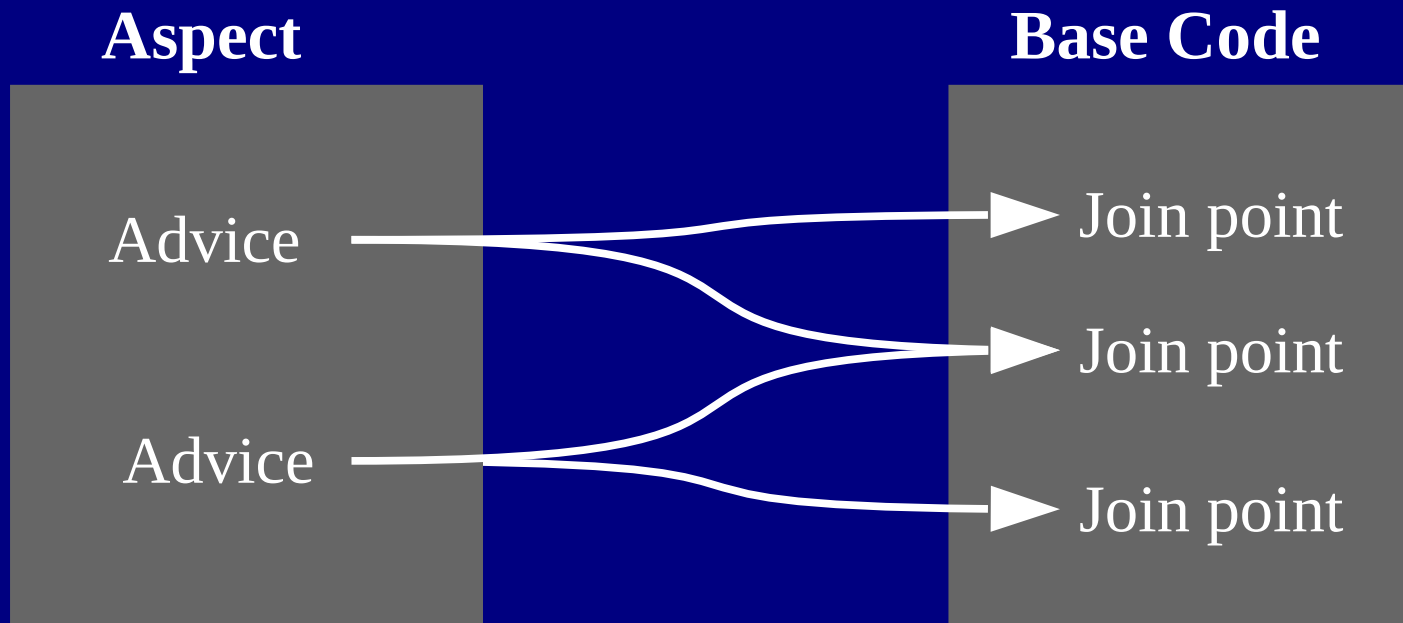


Joint Work With:

**David H. Lorenz**

# Aspect Oriented Programming

“A programming paradigm that aims to increase modularity by allowing the separation of cross-cutting-concerns”



# Known Limitations of AspectJ

- **Rigid join point model**
- **Fragile pointcuts**
- **Imperative syntax for advices**
- **State-point separation issues**

**Base Code**

Join point

Join point



**Base Code**

Join point

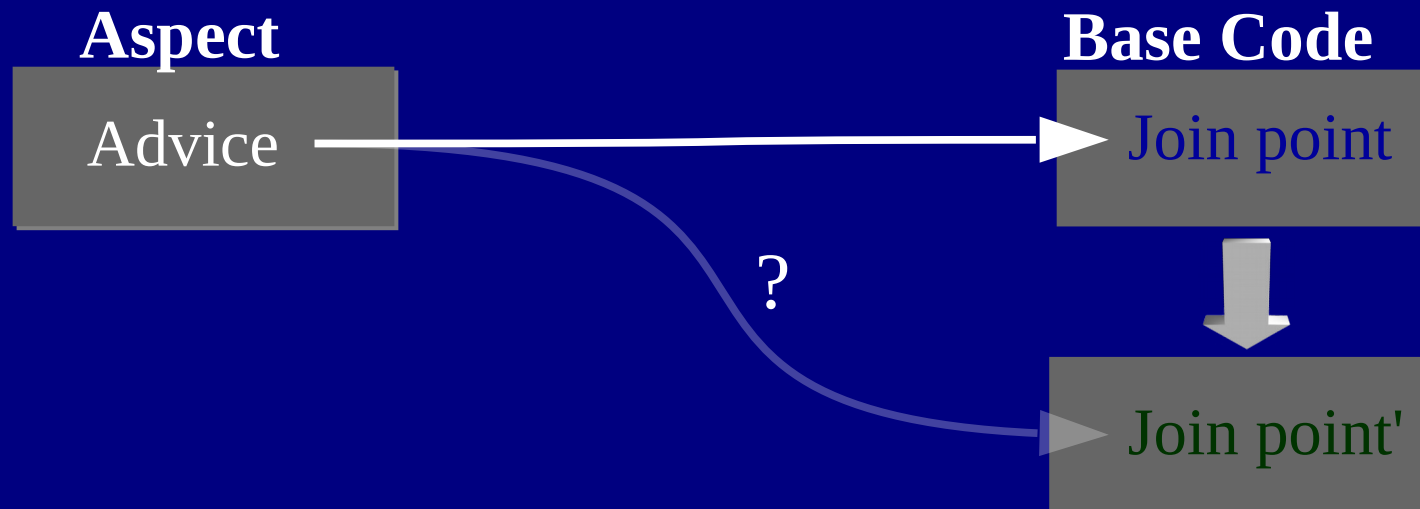
Join point

Join point

Join point

# Known Limitations of AspectJ

- Rigid join point model
- **Fragile pointcuts**
- Imperative syntax for advices
- State-point separation issues



# Known Limitations of AspectJ

- Rigid join point model
- Fragile pointcuts
- Imperative syntax for advices
- State-point separation issues

# Known Limitations of AspectJ

- Rigid join point model
- Fragile pointcuts
- Imperative syntax for advices
- State-point separation issues

# Extensions to AspectJ

- Many extensions were proposed

Closure join  
points

Explicit join  
points

LoopAJ

KALA

COOL

Statement  
annotations

- And more are expected in the future

# Why abc is Not Good Enough

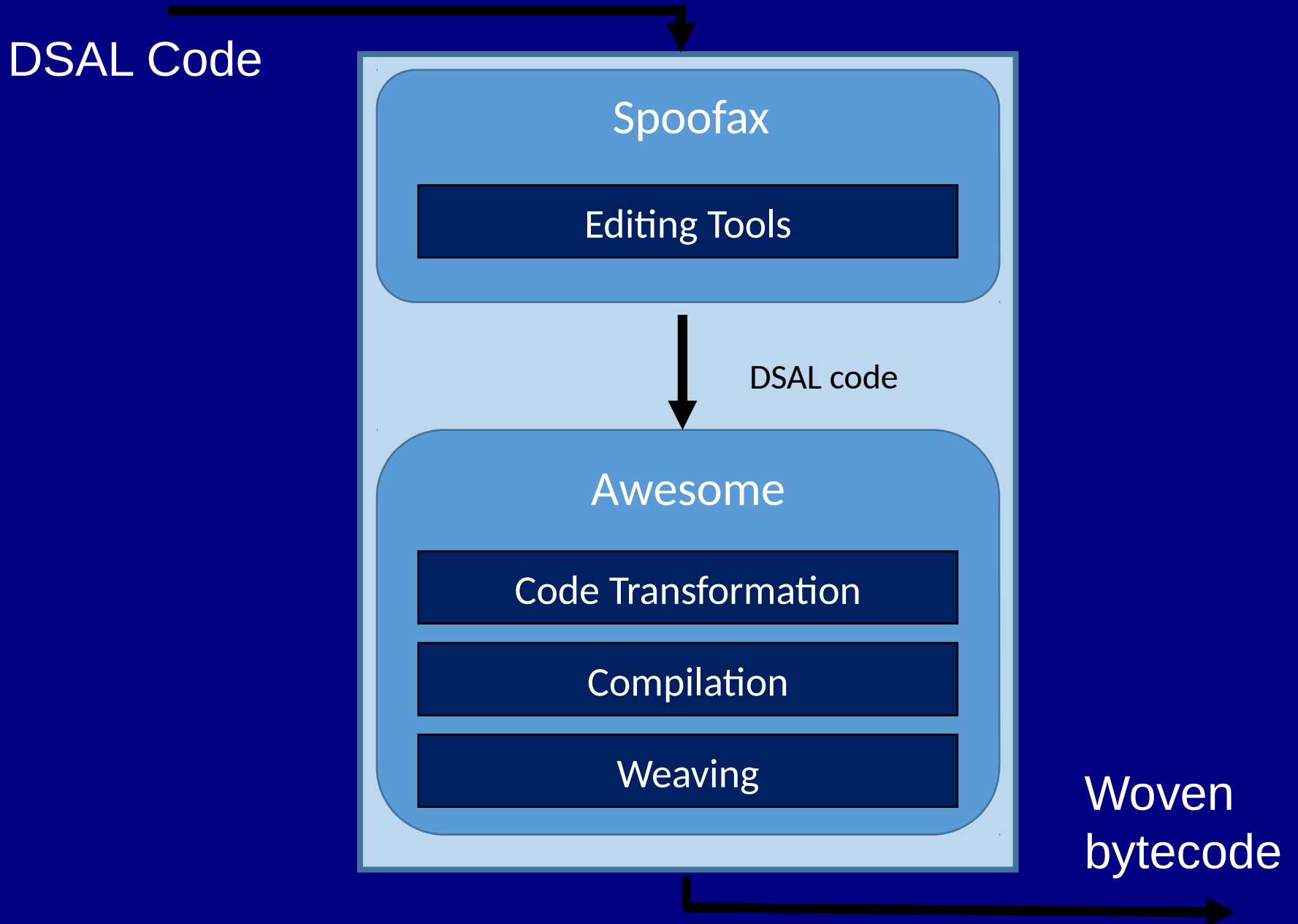
- **Used to be the default choice for implementing AspectJ extensions**
- **Not suitable for development of new extensions**
  - Does not work with recent versions of AspectJ
- **Not suitable for evaluation of new extensions**
  - Does not provide development tools
  - No support for advanced weaving semantics



# Introducing the DSAL Workbench

- **A workbench tailored to AspectJ extensions**
  - Alternative for abc
- **We will present implementations of third-party extensions**
  - Closure Join Points (CJP)
  - Explicit Join Points (EJP)

# The Architecture of Our Workbench



# Grammar Definition of CJP

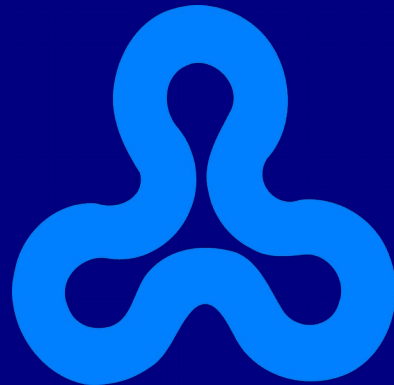
```
Expr ::= ... | ClosureJoinpoint.  
StmtExpr ::= ... | ClosureJoinpoint.  
ClosureJoinpoint ::=  
  "exhibit" ID "(" [ParamList] ")" Block  
  "(" [ArgList] ")" |  
  
  "exhibit" ID Block.  
AspectMember ::= ... | JoinpointDecl.  
JoinpointDecl ::=  
  "joinpoint" Type ID "(" [ParamList] ")" [ThrowsList].  
AdviceDecl ::= ... | CJPAdviceDecl.  
CJPAdviceDecl ::=  
  [Modifiers] CJPAdviceSpec [ThrowsList] Block.  
CJPAdviceSpec ::=  
  Type "before" ID "(" [ParamList] ")" |  
  Type "after" ID "(" [ParamList] ")" |  
  Type "after" ID "(" [ParamList] ")"  
    "returning" [ "(" [Param] ")" ] |  
  Type "after" ID "(" [ParamList] ")"  
    "throwing" [ "(" [Param] ")" ] |  
  Type "around" ID "(" [ParamList] ")".
```

Figure 9: Syntax for Closure Joinpoints, as a syntactic extension to AspectJ (shown in gray)

# Conclusion

- **A DSAL workbench for extensions to AspectJ**
  - Alternative to abc
  - Provides tools to develop and to use DSALs
  - Comprising Spoofox and AWESOME\*
- **Implemented plug-ins for CJP, EJP and COOL**
  - Structured process for creating additional plugins for new extensions
- **The workbench is suitable for the development, evaluation and production of AspectJ extensions**

# Thank You!



**Arik Hadas and David H. Lorenz**

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

[arik.hadas@openu.ac.il](mailto:arik.hadas@openu.ac.il)

<https://github.com/OpenUniversity>