

Advice Activation in AspectS

Robert Hirschfeld
DoCoMo Euro-Labs

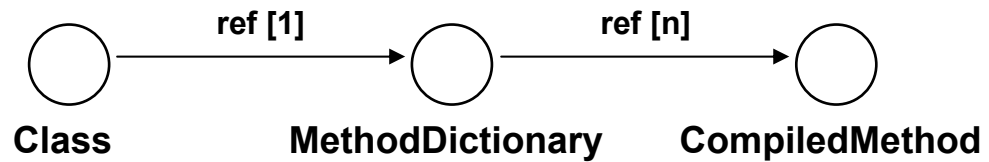
hirschfeld@docomolab-euro.com

February 2002

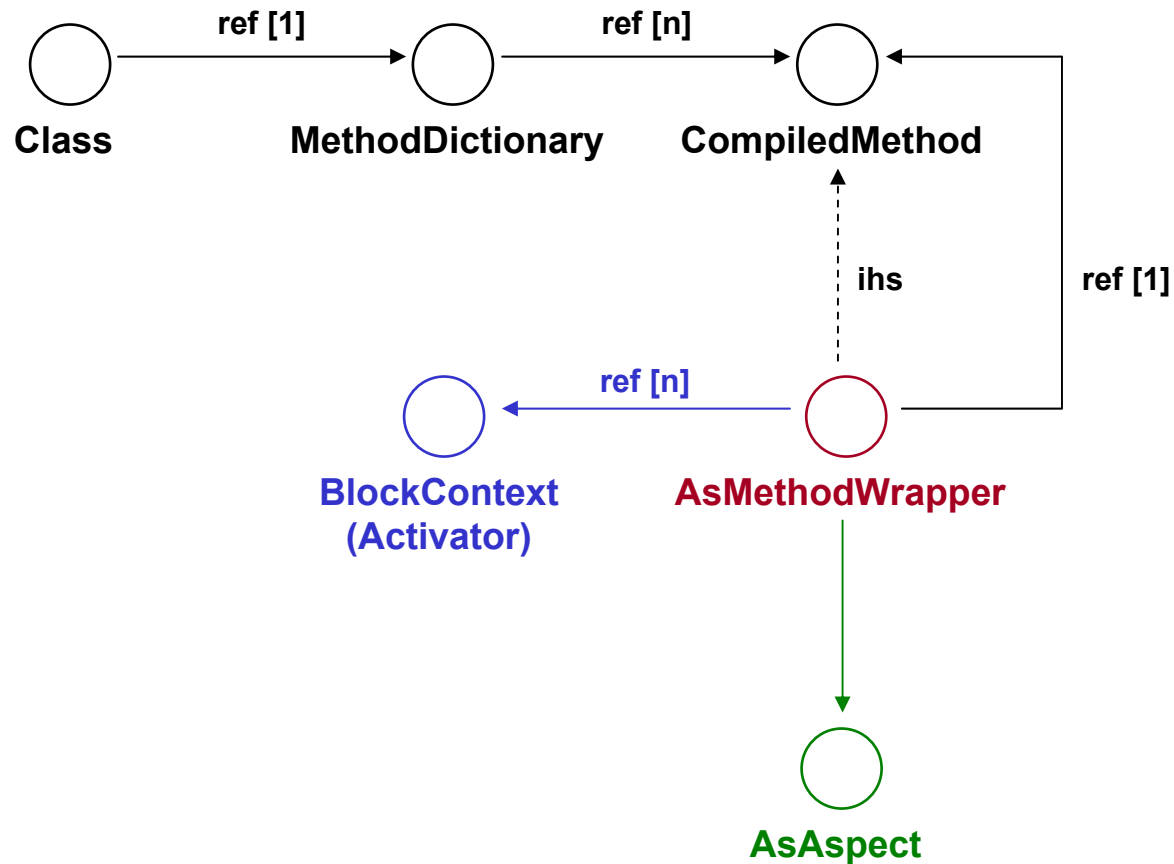
AspectS Overview

- An approach to general-purpose AOP in Squeak/Smalltalk
- Based on concepts of AspectJ (PARC)
- Implemented using Method Wrappers (John Brant)
- Extends the Smalltalk MOP to accommodate the aspect modularity mechanism
- An explorative idea debugger to better understand aspects in a dynamic environment

Smalltalk Messaging



Method Wrappers



AspectS Advice

AsMorphicMousingAspect>>adviceMouseEnter

^ AsBeforeAfterAdvice

qualifier: (AsAdviceQualifier attributes: { #receiverClassSpecific. })

pointcut: [Morph allSubclasses

select: [:each |

each includesSelector: #mouseEnter:]

thenCollect: [:each | AsJoinPointDescriptor

targetClass: each

targetSelector: #mouseEnter:]]

beforeBlock: [:receiver :arguments :aspect :client |

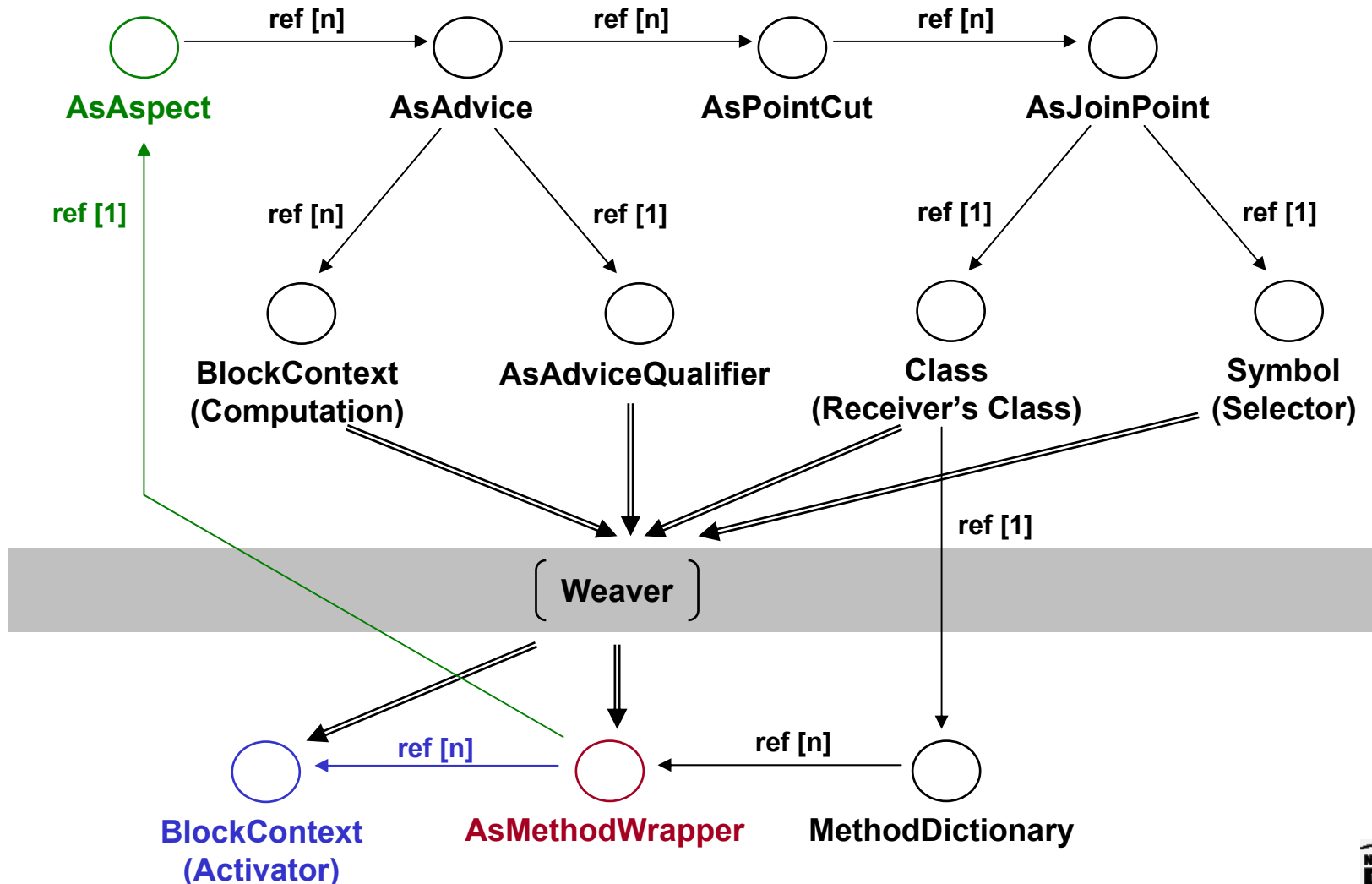
self

showHeader: '>>> MouseENTER >>>'

receiver: receiver

event: arguments first]

AspectS Weaving



Internals

AsMethodWrapper>>isActive

| baseSender |

baseSender := thisContext baseSender.

^ self activators notEmpty

and: [self activators allSatisfy: [:aBlock |

aBlock value: self aspect value: baseSender]]

AsMethodWrapper class>>receiverGeneralActivator

^ [:aspect :baseSender |

true] copy

More Code

AsMethodWrapper class>>cfFirstClassActivator

```
^ [:aspect :baseSender |  
  | lastCfPoint allCfPoints |  
  lastCfPoint := AsCFlowPoint  
    object: baseSender receiver class  
    selector: baseSender selector.  
  allCfPoints := thisContext allBaseClientsWithSelector  
    collect: [:each |  
      AsCFlowPoint  
        object: each key class  
        selector: each value].  
  (allCfPoints occurrencesOf: lastCfPoint) = 1] copy
```

And More...

AsBeforeAfterWrapper>>

valueWithReceiver: anObject arguments: anArrayOfObjects

| client active return |

client := thisContext baseClient.

active := self isActive.

active ifTrue: [self beforeBlock copy valueWithArguments: (Array
with: anObject with: anArrayOfObjects
with: self aspect with: client)].

return := self clientMethod

valueWithReceiver: anObject
arguments: anArrayOfObjects.

active ifTrue: [self afterBlock copy valueWithArguments: (Array
with: anObject with: anArrayOfObjects
with: self aspect with: client with: return)].

^ return