Delegation-Based AOP
A unified execution model for OOP and AOP

Hans Schippers
Formal Techniques in Software Engineering (FoTS)
hans.schippers@ua.ac.be

December 6, 2006

Universiteit Antwerpen
Overview

- Problem Statement
- Virtual Join Points
- Delegation-Based AOP
  - Aspects
  - Threads
  - Control Flow
  - Classes
  - Introductions
- Future Work
- Conclusions
Implementing AOP constructs on top of e.g. JVM is cumbersome

Typical OO VM unfit as execution model for AOP
Virtual Join Points

- JP = point in the execution graph of a running application
- Inherently dynamic
- locus of late-bound functionality dispatch
- similarity to virtual method dispatch
Minimal prototype-based system with delegation (e.g. SELF, Io) suffices as execution model for both OOP and AOP

JP model granularity is at *message* level
Introducing Aspects

Combination of proxy object and explicit delegation (resend)

- Combination of proxy object and explicit delegation (resend)
object’s list of prototypes is a function of the current thread
Control Flow Dimension

- execution of *bar* inserts another proxy temporarily

(a) control flow not yet entered

```
obj
asp_e_cw_proxy
bar = (<activate>, resend, <deactivate>)
```

```
actual_obj
foo = (...)
bar = (..., self foo, ...)
baz = (...)
```

(b) control flow entered in T1

```
obj
asp_e_proxy
foo = (<advice>, resend)
```

```
actual_obj
foo = (...)
```

```
asp_e_cw_proxy
bar = (<activate>, resend, <deactivate>)
```

```
actual_obj
foo = (...., self foo, ...)
baz = (...)
```

(c) control flow entered in T1 and T2

```
obj
asp_e_proxy
foo = (<advice>, resend)
```

```
asp_e_cw_proxy
bar = (<activate>, resend, <deactivate>)
```

```
actual_obj
foo = (...)
```

```
asp_e_proxy
foo = (<advice>, resend)
```

```
t1, t2
asp_e_proxy
foo = (<advice>, resend)
```

```
actual_obj
foo = (...., self foo, ...)
baz = (...)
```

```
obj
asp_e_cw_proxy
bar = (<activate>, resend, <deactivate>)
```

```
actual_obj
foo = (...)
bar = (...., self foo, ...)
baz = (...)
```

```
actual_obj
foo = (...)
bar = (...., self foo, ...)
baz = (...)
```

```
obj
asp_e_cw_proxy
bar = (<activate>, resent, <deactivate>)
```

```
actual_obj
foo = (...)
bar = (...., self foo, ...)
baz = (...)
```
Class object is nothing more than another prototype
Delegation-Based AOP

Supporting Classes (2)

Delegation-Based AOP

- Overview
- Problem Statement
- Virtual Join Points
- Aspects
- Threads
- Control Flow
- Supporting Classes
- Introductions
- Future Work
- Conclusions

Diagram:

```
c1
  asp_h_proxy
  bar = (<advice>, resend)

actual_c1
  x = ...
  y = ...

proxy_C

asphy_proxy
  foo = (<advice>, resend)

actual_c2
  x = ...
  y = ...

c2

C
  foo = (...)
  bar = (..., self foo, ...)
  baz = (...)
```

Delegation-Based AOP
Introductions

(a) nothing introduced yet

(b) an aspect with introductions is deployed

(c) the field f has been introduced to c2
Future Work

- Provide formal semantics
  - Graph Transformation
  - Extend *delta*: object-based calculus with delegation
- VM implementation
Conclusions

- Very simple object-based model with delegation suffices as an execution model for both OOP and AOP
- Better adherence to inherently dynamic definition of a join point
- Will help in the development of dedicated AOP VM’s