

Inductively Generated Pointcuts to Support Refactoring to Aspects

Tom Tourwé (CWI)

Andy Kellens (VUB)

Wim Vanderperren (VUB)

Frederik Vannieuwenhuyse (VUB)

AOSD advantages w.r.t. Evolvability

- Separation of Concerns
 - Different concerns can be evolved separately
- Improved Modularity
 - Reduced propagation of changes

AOSD disadvantages w.r.t. Evolvability

- Evolving an OO application into an AOSD application
 - Often requires refactoring the OO application
- Evolving an AOSD application
 - Often requires changing existing pointcuts

pointcut definition
requires
refactoring



refactoring
requires
pointcut (re)definition

Pointcut Definition

```
class A {  
    public void m() {  
        this.d();  
        x.a();  
        y.b();  
    }  
}
```

Aspect
A

```
class B {  
    public void n() {  
        x.a();  
        y.b();  
        this.e();  
    }  
}
```

- Pointcut: a call to a, followed by a call to b
- Can not be expressed in AspectJ
 - only method executions, method calls, etc

Pointcut Definition

```
class A {  
    public void ab() {  
        x.a();  
        y.b();  
    }  
    public void m() {  
        this.d();  
        this.ab();  
    }  
}
```

Aspect
A

```
class B {  
    public void ab() {  
        x.a();  
        y.b();  
    }  
    public void n() {  
        this.ab();  
        this.e();  
    }  
}
```

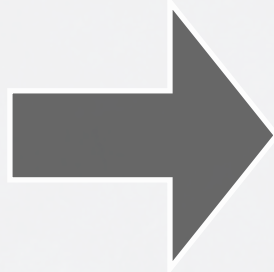
- Refactor to make aspect definition possible!
 - Base code no longer oblivious to aspect
 - Not scalable, not guaranteed to work

Pointcut Refactoring

```
class A {  
    public void m() {}  
}
```

```
aspect X {  
    pointcut pcA():  
        call(void *.m());  
    pointcut pcB():  
        call(void *.n());  
}
```

Rename Method



```
class A {  
    public void n() {}  
}
```

```
aspect X {  
    pointcut pcA():  
        call(void *.m()) || call(A.n());  
    pointcut pcB():  
        call(void*.n()); && !call(A.n());  
}
```

- Tight coupling between pointcuts and program's structure
 - Pointcuts need to be refactored

What if ...?

- ... we consider more complex refactorings?
 - Impact of “base level” refactoring on pointcut not always as clear
- ... we apply several refactorings?
 - Complexity of the resulting pointcut

Primary Causes

- Primitive pointcut languages

- Limited expressiveness

- Pointcuts very tightly coupled to base program's structure

- Limited automated support

- Developer forced to deal with pointcuts at too low a level

More expressive pointcut languages!

High-level support for pointcuts!