

Evolution and Aspect-Oriented Software Development

Tom Tourwé

Centrum voor Wiskunde en Informatica
Amsterdam

Two Main Issues

- **Migration**: evolving “ordinary” (object-oriented) applications into aspect-oriented applications
- **Evolution**: evolving aspect-oriented applications

Migration Issues

- Aspect mining

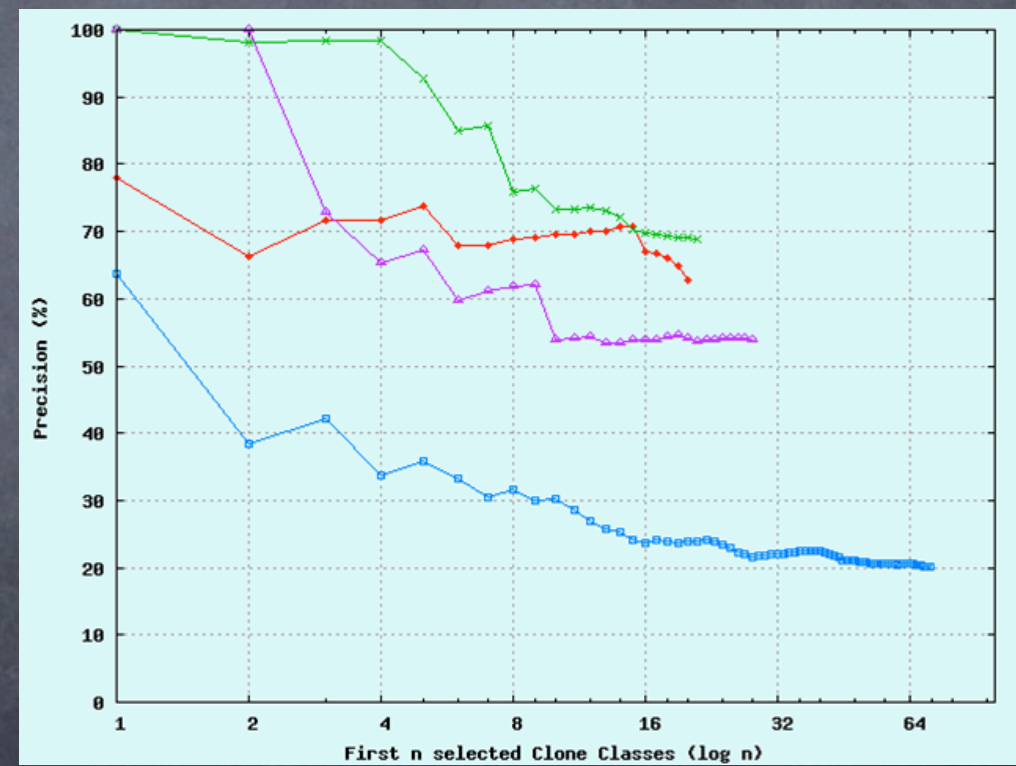
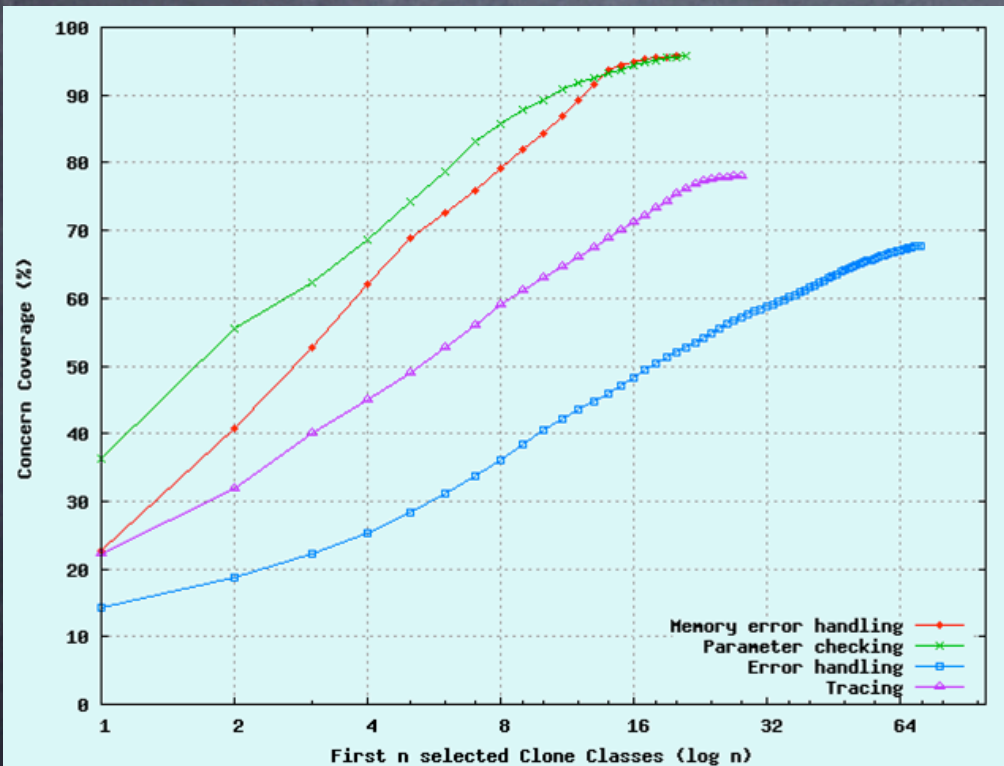
 - identify crosscutting concerns in the source code

- Aspect extracting

 - define aspects that implement the concerns, and remove them from the source code

Migration Experiments

Clone detection as an aspect mining technique?



Migration Experiments

Evaluating benefits of AOSD & feasibility of extracting aspects

Parameter	LoC now	LoC aspects	% reduced
in/out	1170	536	54
pointer	272	58	79
Total	1441	594	59

Parameter	LoC now	LoC aspects	% reduced
input	1115	548	51
output	424	262	38
Total	1539	810	47

Evolution Issues

- Aspect evolution
 - aspect-oriented refactoring?
- Base code evolution
 - aspect-aware refactoring?
- Impact of one on the other and vice versa

Evolution Experiments

Aspect-aware Refactorings?

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

```
aspect X {  
    pointcut pc1():  
        call(void *.m());  
    ...  
    pointcut pc2():  
        call(void *.n());  
    ...  
}
```

Rename
method



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

```
aspect X {  
    pointcut pc1():  
        call(void *.m()) || call(A.n());  
    ...  
    pointcut pc2():  
        call(void *.n()) && !call(A.n());  
    ...  
}
```


Evolution Experiments

Aspect-oriented Refactorings?

```
private String rentalStatement(Rental each) {
    String result = "";
    result += "\t" + each.getMovie().getTitle();
    result += "\t" + each.getCharge() + "\n";
    frequentRenterPoints += each.getFRP();
    totalAmount += each.getCharge();
    return result;
}

private String statementFooter() {
    String result;
    result += "Amount owed is " + totalAmount + "\n";
    result += "You earned " + frequentRenterPoints +
        " frequent renter points";
    return result;
}
```

```
after(Customer c, Rental each):
    call(String rentalStatement(...)) && args(each) &&
    target(c) {
        c.totalAmount += each.getCharge();
    }

String around(Customer c):
    call(String statementFooter(..)) && target(c) {
        String result = proceed(c);
        return result += "Amount owed is " +
            totalAmount + "\n";
    }
```

```
after(Customer c, Rental each):
    call(String rentalStatement(...)) && args(each) &&
    target(c) {
        c.frequentRenterPoints += each.getFRP();
    }

String around(Customer c):
    call(String statementFooter(..)) && target(c) {
        String result = proceed(c);
        return result += "You earned " +
            frequentRenterPoints +
            " frequent renter points";
    }
```


Discussion Topics

Quality
measures

Aspect-oriented
refactorings

Aspect-aware
refactorings

Testability

Behavior
preservation

Aspect mining
techniques

Aspect
languages

BYO