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?

Tom Tourwé - Benevol 2.0 - July, 9th 2004
### Migration Experiments

Evaluating benefits of AOSD & feasibility of extracting aspects

<table>
<thead>
<tr>
<th>Parameter</th>
<th>LoC now</th>
<th>LoC aspects</th>
<th>% reduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>in/out</td>
<td>1170</td>
<td>536</td>
<td>54</td>
</tr>
<tr>
<td>pointer</td>
<td>272</td>
<td>58</td>
<td>79</td>
</tr>
<tr>
<td>Total</td>
<td>1441</td>
<td>594</td>
<td>59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>LoC now</th>
<th>LoC aspects</th>
<th>% reduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>1115</td>
<td>548</td>
<td>51</td>
</tr>
<tr>
<td>output</td>
<td>424</td>
<td>262</td>
<td>38</td>
</tr>
<tr>
<td>Total</td>
<td>1539</td>
<td>810</td>
<td>47</td>
</tr>
</tbody>
</table>
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());
        ...
}

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?

```java
private String rentalStatement(Rental each) {
    String result = "";
    result += "\t" + each.getTitle();
    result += "\t" + each.getCharge() + "\n";
    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;
}
```

```java
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";
    }
```

```java
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";
    }
```

```java
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";
    }
```
Discussion Topics

- Quality measures
- Aspect-oriented refactorings
- Aspect-aware refactorings
- Testability
- Behavior preservation
- Aspect mining techniques
- Aspect languages
- BYO

Tom Tourwé - Benevol 2.0 - July, 9th 2004