A model-driven pointcut language for more robust pointcuts

Andy Kellens  Kim Mens  Johan Brichau  Kris Gybels
Fragile Pointcuts
Fragile Pointcuts

BASE system

Aspect  Aspect  Aspect

Pointcuts in program

Source program
Fragile Pointcuts

BASE system

Aspect  Aspect  Aspect

Pointcuts in program

Pointcuts in evolved program

BASE’ system

Aspect  Aspect  Aspect

Source program

Evolved Source program

Pointcuts in program

EVOLUTION

Aspect

Accidental joinpoint miss

Unintended joinpoint capture
An example

class Buffer {

    private Object content[];
    private int index = 0;

    ...

    public Object get() {
        ... return content[index] ... }

    public void set(Object el) {
        ... content[index] := el ... }
    ...
}
class Buffer {

    private Object content[];
    private int index = 0;
    ...

    public Object get() {
        ... return content[index] ... 
    }

    public void set(Object el) {
        ... content[index] := el ... 
    }
    ...
}
Fragile Pointcuts

Enumeration based

```java
pointcut accessors()
    call(void Buffer.set(..)) ||
    call(Object Buffer.get(..));
```
Fragile Pointcuts

Enumeration based

```java
pointcut accessors()
    call(void Buffer.set(..)) ||
    call(Object Buffer.get(..));
```

Add new getter/setter
Fragile Pointcuts

Enumeration based

```plaintext
pointcut accessors()

call(void Buffer.set(..)) ||
call(Object Buffer.get(..));
```

Pattern based

```plaintext
pointcut accessors()

call(* set*(..)) || call(* get*(..));
```

Add new getter/setter
Fragile Pointcuts

Enumeration based

```java
pointcut accessors()
call(void Buffer.set(..)) ||
call(Object Buffer.get(..));
```

Pattern based

```java
pointcut accessors()
call(* set*(..) ) || call(* get*(..) );
```

Add new getter/setter

Adding setting()
Pointcuts (2)

Structural property-based

pointcut setters
   call(\* ?class.\?method(\_.) ) &&
   assigns(\?method,\?iv) &&
   instanceVariable(\?iv,\?class);

pointcut getters
   call(\* ?class.\?method(\_.) ) &&
   returns(\?method,\?iv) &&
   instanceVariable(\?iv,\?class);
Pointcuts (2)

Structural property-based

**pointcut** setters
- `call(* ?class.?method(..) )` &&
- `assigns(?method,?iv)` &&
- `instanceVariable(?iv,?class)`;

**pointcut** getters
- `call(* ?class.?method(..) )` &&
- `returns(?method,?iv)` &&
- `instanceVariable(?iv,?class)`;

```java
public Object get() {
    Object temp := content[index];
    ...
    return temp;
}
```
Structural property-based

**pointcut setters**

\[
\text{call}(*, \text{?class.}\text{.?method}(..)) \land \text{assigns(?method,?iv)} \land \text{instanceVariable(?iv,?class)};
\]

**pointcut getters**

\[
\text{call}(*, \text{?class.}\text{.?method}(..)) \land \text{returns(?method,?iv)} \land \text{instanceVariable(?iv,?class)};
\]

Fundamental cause: Pointcut relies heavily on how the software is structured at a given moment in time
Fundamental cause

Pointcut tries to deduce implicit concept from the code

Remove fragility through explicit concepts
Model-based pointcuts

Aspect using traditional pointcuts

Source-code pointcut definition

Pointcut in terms of source code

Pointcut in terms of conceptual model

Join point model

Source code
Model-based pointcuts

Aspect using traditional pointcuts

Source-code pointcut definition

Pointcut in terms of source code

Pointcut in terms of conceptual model

Join point model

**pointcut** accessors()

```java
isClassifiedAs(?method, AccessorMethod) && call(?method)
```
Model-based pointcuts

Aspect using traditional pointcuts

Source-code pointcut definition

Aspect using model-based pointcuts

Model-based pointcut definition

Pointcut in terms of source code

Pointcut in terms of conceptual model

Join point model

Classifications and constraints

Conceptual model

Source code

Pointcut accessor() isClassifiedAs(?method, AccessorMethod) && call(?method)
Model-based pointcuts

Aspect using traditional pointcuts

Source-code pointcut definition

Aspect using model-based pointcuts

Model-based pointcut definition

Pointcut in terms of source code

Pointcut in terms of conceptual model

Join point model

Classifications and constraints

Conceptual model

Source code

Pointcut in terms of model

pointcut accessors()

isClassifiedAs(?method, AccessorMethod) && call(?method)
Model-based pointcuts

Aspect using traditional pointcuts

Source-code pointcut definition

Aspect using model-based pointcuts

Model-based pointcut definition

Pointcut in terms of source code

Pointcut in terms of conceptual model

Join point model

Classifications and constraints

Conceptual model

Source code

Pointcut in terms of model

Keep model consistent

pointcut

isClassifiedAs(?method, AccessorMethod) && call(?method)
Implementation

Requires:

- Conceptual model
  - in terms of source code
  - ability to specify constraints
  - remain consistent with code
- Pointcut language constructs
# Implementation

Requires:

- Conceptual model
  - in terms of source code
  - ability to specify constraints
  - remain consistent with code
- Pointcut language constructs
## Implementation

Requires:

- Conceptual model in terms of source code
- Ability to specify constraints
- Remain consistent with code
- Pointcut language constructs
Co-evolving code and design using Intensional Views - A Case Study
Kim Mens, Andy Kellens, Frederic Pluquet, Roel Wuyts
Co-evolving code and design using Intensional Views - A Case Study
Kim Mens, Andy Kellens, Frederic Pluquet, Roel Wuyts
Intensional Views

Accessors

set* || get*

structural property

Specified intensionally

Buffer.get()

Buffer.set(…)

Co-evolving code and design using Intensional Views - A Case Study
Kim Mens, Andy Kellens, Frederic Pluquet, Roel Wuyts
Co-evolving code and design using Intensional Views - A Case Study
Kim Mens, Andy Kellens, Frederic Pluquet, Roel Wuyts
Co-evolving code and design using Intensional Views - A Case Study
Kim Mens, Andy Kellens, Frederic Pluquet, Roel Wuyts
Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts

Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels

Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts
Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels
### Experiment

<table>
<thead>
<tr>
<th>Time</th>
<th>SmallWiki</th>
<th>Traditional</th>
<th>Model-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>1.54</td>
<td>Logging</td>
<td>Logging</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>Output</td>
</tr>
<tr>
<td>2003</td>
<td>1.304</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts

Andy Kellens, Kim Mens, Johan Brachau, Kris Gybels

*In "Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP) (to appear)", 2006*
Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts
Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels
Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts
Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels
Discussion

Language integration of tool support

• Independent of pointcut language and conceptual model

• Pointcuts no longer expressed in terms of program structure

• Does not solve fragile pointcuts; rather detect and resolve

• No certainty that everything will be detected

• Need for a methodology
• More information:
  • http://www.intensional.be
  • http://prog.vub.ac.be/carma/
  • Demo this thursday at 10:30 and 14:00