Problem

Aspect Orientation  →  Better modularization
Problem

Aspect Orientation  Better modularization
Software easier to evolve
Problem

Aspect Orientation ——— Better modularization
Software easier to evolve

Evolution Paradox
  • High coupling
  • Fragile pointcuts
Fragile Pointcuts

Legend
- Specified in terms of:
  - Captured join point
  - Uncaptured join point

"Fragile" pointcut

Pointcuts in program

Pointcuts in evolved program

Evolution

Accidental joinpoint miss

Unintended joinpoint capture
Examples

Accessors

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

Accessors

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

**Accessors**

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

**Accessors**

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

**Addition of any setter/getter**
Examples

Accessors

\[
\text{call(void Buffer.set(..)) } \lor \text{ call(Object Buffer.get(..))};
\]

Accessors

\[
\text{call(* set*(..) ) } \lor \text{ call(* get*(..) )};
\]

Addition of any setter/getter

Addition of eg. settings()
Examples(2)

Accessors

Setters

call(* ?class.?method(..) ) && assigns(?method,?iv) && instanceVariable(?iv,?class);

Getters

call(* ?class.?method(..) ) && returns(?method,?iv) && instanceVariable(?iv,?class);

Optimized getters

getters() && not(cflow(getters()));
Examples(2)

Accessors

Setters

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

Getters

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

Optimized getters

\[
\text{getters}() \ && \ 
\text{not}(\text{cflow}(\text{getters}()));
\]
Examples(2)

Accessors

Setters

call(* ?class.?method(..) ) &&
assigns(?method,?iv) &&
instanceVariable(?iv,?class);

Getters

call(* ?class.?method(..) ) &&
returns(?method,?iv) &&
instanceVariable(?iv,?class);

Optimized getters

getters() &&
not(cflow(getters()));

{  
    Object temp := content[index];
    ...
    return temp;
}

Ok, but getters() is in function of source-code
MODEL-BASED POINTCUTS

Conceptual Model

Source code

Evolved Source code

Model-based pointcut

Pointcuts

Pointcuts in Evolved Code

EVOLUTION
Model-based Pointcuts

Accessors

isClassifiedAs(?method,Accessor Method) && call(?method)

Requires:

- Conceptual Model
  - In function of implementation
  - Possibility to define constraints
  - Remain consistent with implementation
- Pointcut Language constructs
Model-based Pointcuts

Accessors

\[
\text{isClassifiedAs(}\text{?method,Accessor Method)} \land \text{call(}\text{?method)}
\]

Requires:

- Conceptual Model
  - In function of implementation
  - Possibility to define constraints
  - Remain consistent with implementation

- Pointcut Language constructs

Intensional Views
### Model-based Pointcuts

**Accessors**

\[
\text{isClassifiedAs}(\text{?method}, \text{Accessor Method}) \land \text{call}(\text{?method})
\]

**Requires:**

<table>
<thead>
<tr>
<th>Conceptual Model</th>
<th>Intensional Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>In function of implementation</td>
<td></td>
</tr>
<tr>
<td>Possibility to define constraints</td>
<td></td>
</tr>
<tr>
<td>Remain consistent with implementation</td>
<td></td>
</tr>
</tbody>
</table>

| Pointcut Language constructs                               | CARMA                                                 |
**Model-based Pointcuts**

**Accessors**

\[
\text{isClassifiedAs(?method,Accessor Method) \&\& call(?method)}
\]

**Requires:**

- Conceptual Model
  - In function of implementation
  - Possibility to define constraints
  - Remain consistent with implementation

- Pointcut Language constructs

**View-based pointcuts**

- Intensional Views
  - CARMA
Intensional Views

- Intensional View
  - Set of source-code entities
  - Defined by means of an executable intension

- Constraints:
  - Multiple, alternative intensions
  - Binary relations between views

- Verifiable:
  - Alternatives should yield same set of source-code entities
  - Checking relations with respect to source code
  - Providing fine-grained feedback in order to support co-evolution
Model of SmallWiki

Modeled design structure of SmallWiki 1.54 (previous Benevol)

SmallWiki Actions

Alternative 1

\[
\text{classInNamespace(}\text{?class,SmallWiki},
\text{methodWithNameInClass(}\text{?entity,?name,?class},
\text{[‘execute*’ match: ?name]}
\]

Alternative 2

\[
\text{classInNamespace(}\text{?class,SmallWiki},
\text{methodInClass(}\text{?entity,?class},
\text{methodInProtocol(}\text{?entity,action)}
\]

\[\forall \ x \in \text{SmallWiki Actions: } \exists \ y \in \text{Action Handler: } x \ \text{isImplementedBy} \ y\]
We want to log all executions of Actions in SmallWiki

Two possible pointcuts:

- **SmallWiki Actions**
  
  ```
  classInNamespace(?class,[SmallWiki]),
  methodWithNameInClass(?method,?selector,?class),
  ['execute*' match: ?selector],
  reception(?joinpoint,?selector,?arguments)
  ```

- **SmallWiki Actions**
  
  ```
  classInNamespace(?class,[SmallWiki]),
  methodWithNameInClass(?method,?selector,?class),
  methodInProtocol(?method,action),
  reception(?joinpoint,?selector,?arguments)
  ```
Evolving logging aspect

Re-apply the aspect to version 1.90 of SmallWiki (one year after 1.54)

SmallWiki Actions

classInNamespace(?class,[SmallWiki]),
methodWithNameInClass(?method,?selector,?class),
['execute*' match: ?selector],
reception(?joinpoint,?selector,?arguments)
Evolving logging aspect

Re-apply the aspect to version 1.90 of SmallWiki (one year after 1.54)

SmallWiki Actions

- classInNamespace(?class,[SmallWiki]),
- methodNameInClass(?method,?selector,?class),
- ['execute*' match: ?selector],
- reception(?joinpoint,?selector,?arguments)

Addition of 'save' and 'authenticate'

SmallWiki Actions

- classInNamespace(?class,[SmallWiki]),
- methodNameInClass(?method,?selector,?class),
- methodInProtocol(?method,action),
- reception(?joinpoint,?selector,?arguments)
Evolving logging aspect

Re-apply the aspect to version 1.90 of SmallWiki (one year after 1.54)

SmallWiki Actions

classInNamespace(?class,[SmallWiki]),
methodWithNameInClass(?method,?selector,?class),
[‘execute*’ match: ?selector],
reception(?joinpoint,?selector,?arguments)

SmallWiki Actions

classInNamespace(?class,[SmallWiki]),
methodWithNameInClass(?method,?selector,?class),
methodInProtocol(?method,action),
reception(?joinpoint,?selector,?arguments)

Addition of ‘save’ and ‘authenticate’

Addition of executeSearch and executePermission (part of protocol ‘private’)

Model-based pointcut

SmallWiki Actions

classifiedAs(?method,'SmallWiki Actions'),
methodWithName(?method,?message),
isReceptionOfWith(?joinpoint,?message,?args),
withinClass(?joinpoint,?class)

classInNamespace(?class,SmallWiki),
methodWithNameInClass(?entity,?name,?class),
[‘execute*’ match: ?name]

Alternative 2

classInNamespace(?class,SmallWiki),
methodInClass(?entity,?class),
methodInProtocol(?entity,action)

∀ x ∈ SmallWiki
Actions: ∃ y ∈ Action
Handler: x
isImplementedBy y
Evolving model-based pointcut

SmallWiki Actions

- classifiedAs(?method,'SmallWiki Actions'),
- methodWithName(?method,?message),
- isReceptionOfWith(?joinpoint,?message,?args),
- withinClass(?joinpoint,?class)

- Check consistency of views and relations
  - ‘Save’ and ‘authenticate’ are covered by second alternative but not by first
  - ‘executeSearch’ and ‘executePermission’ are covered by first alternative but not by second

- Update views and/or code

- No need to update pointcut
Discussion

- Independent of chosen pointcut language and technique to model concepts
- Pointcut no longer expressed in terms of source code; less brittle
- Does not prevent the fragile pointcut problem, instead tries to detect inconsistencies
- Not certain that all inconsistencies are detected; depends on the conceptual model and the constraints
- Need methodology; possible candidate “Design Rules” [Sullivan et al.]
Questions