Aspectual Feature Modules

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo
Aspects vs. Feature Modules

- Aspects and feature modules are powerful mechanisms
  - They have different strengths and weaknesses

- We need guidelines about when to use which mechanism
  - Programmers should use the best of the two worlds
Comparison in Two Dimensions

- Homogeneous or Heterogeneous
  - They depend on the uniformity of the program extensions

- Static or Dynamic
  - They depend on the temporal pattern of extensions
Homogeneous and Heterogeneous Concerns
Definitions

- A homogeneous crosscutting concern extends a program at multiple join points by applying a single extension.

- A heterogeneous crosscutting concern extends multiple join points by adding multiple extensions.
  - Each extension targets one join point.
Examples

Homogeneous extension

```java
class A {
    public void foo() {
        // do something
    }
}
```

```java
class B {
    public void foo() {
        // do something
    }
}
```

```java
class C {
    public void foo() {
        // do something
    }
}
```

Heterogeneous extension

```java
class A {
    public void foo() {
        // do something
    }
}
```

```java
class B {
    public void foo() {
        // do something
    }
}
```

```java
class C {
    public void foo() {
        // do something
    }
}
```

```java
public aspect FooAspect {
    after(): execution(void A.foo()) ||
    execution(void B.foo()) ||
    execution(void C.foo()) {
        // do something
    }
}
```

```java
public aspect FooAspect {
    after(): execution(void A.foo()) ||
        // do something
}
```

```java
...```
When to use heterogeneous?

- Collaborations of classes are typically heterogeneous
  - Classes add to a program different functionality

- Feature modules are also designed to implement heterogeneous concerns
When to use homogeneous?

- Aspects perform better in extending a set of join points using a single advice
  - That is, modularizing a homogeneous crosscutting concern
  - Aspects avoid code replication

- If feature-oriented programming is applied to homogeneous crosscutting concerns, refinements may introduce the same code (replications)
Static and Dynamic Concerns
Definitions

- A static crosscutting concern extends the structure of a program statically
  - It adds classes, interfaces, fields, and methods

- A dynamic crosscutting concern affects the control flow of a program
  - It runs additional code when predefined events occur during the program execution
Examples of Static Concern

Aspect-oriented programming (AOP)

```java
aspect ComparableEdge {
    declare parents: Edge implements Comparable;
    Boolean Edge.compare(Edge e) {
        // ...
    }
}
```

Feature-oriented programming (FOP)

```java
refines class Edge implements Comparable {
    boolean compare(Edge e) {
        // ...
    }
}
```
Examples of Dynamic Concern

Aspect-oriented programming (AOP)

```java
aspect Weighted {
    after(Edge e): execution(void Edge.print())
        && this(e) {
            e.weight.print();
        }
}
```

Feature-oriented programming (FOP)

```java
refines class Edge {
    void print() {
        Super.print();
        weight.print();
    }
}
```
Comparison

- Both aspects and feature modules can extend the structure of a base program statically.

- Feature-oriented programming does not provide language support for dynamic concerns.
  - In some cases, they can be implemented.
## Summary of Comparison

<table>
<thead>
<tr>
<th>Crosscutting</th>
<th>FOP</th>
<th>AOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous</td>
<td>Good Support</td>
<td>Limited Support</td>
</tr>
<tr>
<td>Homogeneous</td>
<td>No Support</td>
<td>Good Support</td>
</tr>
<tr>
<td>Static</td>
<td>Good Support</td>
<td>Limited Support</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Weak Support</td>
<td>Good Support</td>
</tr>
</tbody>
</table>
Combining Aspects and Feature Modules
Combining Approaches

- AOP and FOP are not competing approaches
  - They can be combined

- A program can be decomposed in three dimensions
  - Classes and interfaces
  - Features (refinements)
  - Aspects
Feature modules that contain aspects and refinements

Example:
- Section 6.3