Using Design Structure Matrices to Assess Modularity in Aspect-Oriented Software Product Lines

Pedro Matos Jr.
Rafael Duarte
Ivan Cardim
Paulo Borba
Software Product Lines

- Core features + specific set of features for each product
- Our case study:
  - Mobile Games
SPL & Aspects

- Aspects are a very interesting way of implementing variability
  - Each variation is represented by one aspect
  - Variations are chosen during build time
Aspects are good for SPL?

- How to deal with variability in software product lines?
  - Aspects

- Is it a modular way to solve this problem?

- Let’s use Design Structure Matrices to check it
Modularity

- How the use of aspects to implement variability impacts on the modularity?

- At first glance, it seems to improve it...
Case Study

- BestLap
  - Mobile game developed in J2ME by Meantime
  - +20 different phones supported
  - Two different ways to implement variations:
    - Conditional compilation
    - Aspects
What We Have Found Out?

- The DSMs helped us to find a strong dependence between base code and aspects.
- There is a cyclical dependence between base code and the aspects used to implement the variations.

<table>
<thead>
<tr>
<th>package</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>game.screen.128x128</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>game.screen.128x160</td>
<td></td>
<td>2</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>game.screen.128x117</td>
<td></td>
<td></td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>game.gui</td>
<td>4</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Cyclical dependences are not the only problem

- **game.gui** package depends on all screen size packages
  - This package should depend only on an abstract concern **screen size**, not on every possible implementation for this concern
Possible Solution

- Introduce Design Rules to avoid cyclical dependencies and allow parallel development
- Some kind of class-aspect interface which could deal with the variations

<table>
<thead>
<tr>
<th>package</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>b.dr.screen</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>game.screen.128x128</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>game.screen.128x160</td>
<td>3</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>game.screen.128x117</td>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>game.gui</td>
<td>5</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- AOP is a really good choice for SPL, but it still needs new ways of improving its modularity
- DSMs and design rules provided a good insight on where new constructs could be focused
References


Using Design Structure Matrices to Assess Modularity in Aspect-Oriented Software Product Lines

Pedro Matos Jr.
Rafael Duarte
Ivan Cardim
Paulo Borba