Extending a Taxonomy of Bad Code Smells with Metrics

R Marticorena, C Lopez, The University of Burgos
Y Crespo, University of Valladolid

Presented by D Rodríguez, The University of Reading
Introduction

- Refactoring
- Bad Smells, i.e., collect design or code flaws
- but ...
  - ...their detection is based on certain programmer intuition
  - current taxonomies do not settle relations between bad smell groups and used metrics
    - Until now, grouping bad smells depending of certain similarity, more or less subjective
Related Works

- Fowler’s Refactoring book.
  - “no set of metrics rivals human intuition”
  - but with large amounts of code, intuition can be difficult

- Taxonomies
  - “within classes” and “between classes”.
  - [http://wiki.java.net/bin/view/People/SmellsToRefactorings](http://wiki.java.net/bin/view/People/SmellsToRefactorings)
  - Mantyla -> based on similarity criteria:
    - Bloaters, Object-Oriented Abusers, Change Preventers, Dispensables and Couplers.
    - Also, metrics for detecting bad smells

- Muñoz -> queries on logic predicates
- Marinescu -> queries on a database
Extended Taxonomy of Bad Smells

- Extending and Crossing Mantyla taxonomy with metrics:
  - Granularity - size of the component. In OO:
    - following levels: system class and method.
  - Intra vs. Inter-relations (Intra) the bad smell could be observed from the individual observation (intra) of the component
  - Inheritance (IH) information about inheritance hierarchy is needed to suggest the bad smells.
  - Access Modifiers (Acc) access level among the components.

- Crossing
  - With software metrics and specially OO metrics
### Example

<table>
<thead>
<tr>
<th>Bad Smell</th>
<th>Group</th>
<th>Gran.</th>
<th>Intra</th>
<th>IH</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Clumps</td>
<td>Bloater</td>
<td>Class</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Large Class</td>
<td>Bloater</td>
<td>Class</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Long Method</td>
<td>Bloaters</td>
<td>Method</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Long Parameter List</td>
<td>Bloaters</td>
<td>Method</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Primitive Obsession</td>
<td>Bloaters</td>
<td>Method</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Alternative Classes with Diffe-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Desc.</th>
<th>Gran.</th>
<th>Intra</th>
<th>Inh</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOC</td>
<td>Comment Lines of Code</td>
<td>Class</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>V(G)</td>
<td>McCabe’s Cyclomatic Complexity</td>
<td>Method</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>NP</td>
<td>Number of Parameters</td>
<td>Method</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>DIT</td>
<td>Depth of Inheritance Tree</td>
<td>Class</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>RFC</td>
<td>Response for Class</td>
<td>Class</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
which tool can help us to suggest the presence of a greater number of bad code smells?

- Metrics-1.3.6, RefactorIt 2.5, DMS

- RefactorIt is the most adequate to detect bad code smells
Conclusions and Future Work

- Presented an extension to current taxonomies.
- Metrics are useful for detecting bad code smells.
- Decision system that can be improved with heuristics in future works.