EXPERIMENTAL ASSESSMENT OF SOFTWARE METRICS USING AUTOMATED REFACTORING

Mel Ó Cinnéide*, Laurence Tratt‡, Mark Harman†, Steve Counsell¥, and Iman Hemati Moghadam†

* University College Dublin, Ireland,
‡ King’s College London, UK,
† University College London, UK,
¥ Brunel University, UK.
ROADMAP

- Introduction and Motivation
- Experimental Approach
- Code-Imp: our Refactoring Platform
- Experimental Results
- Conclusion
THE BEWILDERING WORLD OF SOFTWARE METRICS

DCC
ICP
ANA
COH
WMC
ICBMC
DSC
CAMC
CIS
SCOM
ICH
RFC
DAC
NOH
CBO
DCC
CSP
CF
LCOM4
LCOM3
NOH
CBO
DCC
DCC
DCC
DAC
CPCC
AIF
TCC
MIF
IIF
LCOM5
LCOM1
LCC
LSCC
LSCC
CPCC
LCOM4
ICBMC
ICBMC
ICBMC
ICBMC
ICBMC
CIS
CIS
CIS
CIS
CIS
CIS
CIS
CC
SNHD
NOM
NHS
CAI
CBMC
CCDA
AHF
ICBMC
MOA
DAM
RFC
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COM
COMM
Analytic approaches have limitations

- Comparing formulae isn’t easy:

\[ CC(c) = 2 \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{|I_i \cap I_j|}{|I_i \cup I_j|} / k(k - 1) \]

\[ LCOM5(c) = \frac{k - 1}{l} \sum_{a \in A_I(c)} |\{ m | m \in M_I(c) \land a \in I_m \}| / k - 1 \]

- and may not tell us much about the practical aspects of the metric.
Our goal is to **animate** the metrics and make them agents of change.

We use refactorings to change the metrics.
Refactoring and Metrics: an Observation

Refactoring typically has an impact on metrics.

By calculating metric values before and after applying refactoring R, we observe the behaviour of metrics and learn how they compare with other.

<table>
<thead>
<tr>
<th>metric</th>
<th>P₀</th>
<th>P₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric₁</td>
<td>1.23</td>
<td>1.86</td>
</tr>
<tr>
<td>metric₂</td>
<td>78.3</td>
<td>62.8</td>
</tr>
</tbody>
</table>
CODE-IMP: A FRAMEWORK FOR SEARCH-BASED REFACTORING
**IMPLEMENTED TOOL: CODE-IMP**

- An automated search-based refactoring framework

- Three aspects to the refactoring that takes place
  - The set of refactorings that can be applied
  - The type of search technique employed
  - The fitness function that directs the search
CODE-IMP REFACTORINGS

- **Method-level refactorings**
  - Push Down / Pull Up Method
  - Decrease/Increase Method Accessibility

- **Field-level refactorings**
  - Push Down / Pull Up Field
  - Decrease/Increase Field Accessibility

- **Class-level refactorings**
  - Extract/Collapse Hierarchy
  - Make Superclass Abstract/Concrete
  - Replace Inheritance with Delegation
  - Replace Delegation with Inheritance
The Refactoring Process

- Metrics are read after each refactoring is applied

<table>
<thead>
<tr>
<th>metric1</th>
<th>P_0</th>
<th>P_1</th>
<th>P_2</th>
<th>P_3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.12</td>
<td>2.67</td>
<td>2.89</td>
<td>2.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>metric2</th>
<th>P_0</th>
<th>P_1</th>
<th>P_2</th>
<th>P_3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.73</td>
<td>8.52</td>
<td>8.66</td>
<td>8.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.7</td>
</tr>
</tbody>
</table>
INVESTIGATION I: GENERAL ASSESSMENT OF COHESION METRICS
In this investigation we explore five popular cohesion metrics:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSCC</td>
<td>Low-level Similarity-Based Class Cohesion</td>
<td>Al Dallal and Briand, 2010</td>
</tr>
<tr>
<td>CC</td>
<td>Class Cohesion</td>
<td>Bonja and Kidanmariam, 2006</td>
</tr>
<tr>
<td>SCOM</td>
<td>Sensitive Class Cohesion</td>
<td>Fernández and Peña, 2006</td>
</tr>
<tr>
<td>LCOM5</td>
<td>Lack of Cohesion between Methods</td>
<td>Henderson-Sellers, 1996</td>
</tr>
<tr>
<td>TCC</td>
<td>Tight Class Cohesion</td>
<td>Biemann and Kang, 1995</td>
</tr>
</tbody>
</table>
SOFTWARE ANALYSED

- We analysed over 300,000 lines of Java code.

<table>
<thead>
<tr>
<th>Application</th>
<th># LOC</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArtOfIllusion</td>
<td>87,352</td>
<td>459</td>
</tr>
<tr>
<td>JabRef</td>
<td>61,966</td>
<td>675</td>
</tr>
<tr>
<td>JGraphX</td>
<td>48,810</td>
<td>229</td>
</tr>
<tr>
<td>GanttProject</td>
<td>43,913</td>
<td>547</td>
</tr>
<tr>
<td>XOM</td>
<td>28,723</td>
<td>212</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>14,577</td>
<td>208</td>
</tr>
<tr>
<td>JRDF</td>
<td>12,773</td>
<td>206</td>
</tr>
<tr>
<td>JTar</td>
<td>9,010</td>
<td>59</td>
</tr>
</tbody>
</table>
Fitness Function

- Our goal is to explore the metrics, not to improve the program being refactored.

- Applying refactorings randomly will usually cause all metrics to deteriorate.

- So we apply the first refactoring we find that improves at least one of the metrics.

- We measured:
  1. Volatility
  2. Probability of positive change
Volatility is dependent on a combination of a metric and the application to which it is applied (and also on the applied refactorings).
EXPERIMENT AND RESULTS

<table>
<thead>
<tr>
<th>Application</th>
<th>N</th>
<th>LSCC</th>
<th>TCC</th>
<th>SCOM</th>
<th>CC</th>
<th>LCOM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>JHotDraw</td>
<td>1007</td>
<td>50↑ 46↓</td>
<td>45↑ 41↓</td>
<td>38↑ 40↓</td>
<td>53↑ 47↓</td>
<td>51↑ 49↓</td>
</tr>
<tr>
<td>XOM</td>
<td>193</td>
<td>57↑ 43↓</td>
<td>51↑ 46↓</td>
<td>50↑ 44↓</td>
<td>51↑ 49↓</td>
<td>48↑ 52↓</td>
</tr>
<tr>
<td>ArtOfIllusion</td>
<td>593</td>
<td>57↑ 42↓</td>
<td>52↑ 35↓</td>
<td>44↑ 33↓</td>
<td>58↑ 42↓</td>
<td>56↑ 43↓</td>
</tr>
<tr>
<td>GanttProject</td>
<td>750</td>
<td>53↑ 43↓</td>
<td>39↑ 31↓</td>
<td>40↑ 40↓</td>
<td>57↑ 42↓</td>
<td>50↑ 50↓</td>
</tr>
<tr>
<td>JabRef</td>
<td>257</td>
<td>54↑ 46↓</td>
<td>34↑ 27↓</td>
<td>37↑ 42↓</td>
<td>55↑ 44↓</td>
<td>49↑ 50↓</td>
</tr>
<tr>
<td>JRDF</td>
<td>13</td>
<td>46↑ 46↓</td>
<td>23↑ 23↓</td>
<td>46↑ 46↓</td>
<td>46↑ 46↓</td>
<td>54↑ 46↓</td>
</tr>
<tr>
<td>JTar</td>
<td>115</td>
<td>50↑ 49↓</td>
<td>30↑ 23↓</td>
<td>34↑ 36↓</td>
<td>52↑ 46↓</td>
<td>50↑ 40↓</td>
</tr>
<tr>
<td>JGraph</td>
<td>525</td>
<td>51↑ 48↓</td>
<td>37↑ 35↓</td>
<td>36↑ 53↓</td>
<td>61↑ 39↓</td>
<td>41↑ 59↓</td>
</tr>
</tbody>
</table>

Spearmann rank correlation between the metrics

<table>
<thead>
<tr>
<th></th>
<th>LSCC</th>
<th>TCC</th>
<th>SCOM</th>
<th>CC</th>
<th>LCOM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCC</td>
<td>0.60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOM</td>
<td>0.70</td>
<td>0.58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>0.10</td>
<td>0.01</td>
<td>-0.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM5</td>
<td>-0.17</td>
<td>-0.21</td>
<td>-0.46</td>
<td></td>
<td>0.72</td>
</tr>
</tbody>
</table>
We categorise each metric pair as follows:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>Both metrics improve, disimprove, or remain the same</td>
</tr>
<tr>
<td>Dissonance</td>
<td>One metric changes while the other remains the same</td>
</tr>
<tr>
<td>Conflicted</td>
<td>One metric improves while the other disimproves</td>
</tr>
</tbody>
</table>

45% agreement, 17% dissonance, and 38% conflict

The conflicted figure indicates that the metrics embody contradictory notions of cohesion -- a unified notion of cohesion is impossible.
INVESTIGATION II:
COMPARISON OF TCC VS. LSCC
AN ANALYSIS OF TCC VS. LSCC

- In Investigation II we show how our approach can be used to compare two metrics in detail.

- Our aim is to have a qualitative and quantitative analysis of TCC VS. LSCC and more specifically investigate the effect of including inheritance in the metrics definition.

- A single application is refactored, JHotDraw.
Fitness Function

- A refactoring is accepted only if it is Pareto optimal across all the classes of the application.

- We expect that a refactoring that fulfills this robust criterion is likely to be acceptable to a programmer.
EXPERIMENTS AND RESULTS

- To inherit or not to inherit

![Graphs showing correlation between TCC, LSCC, TCCi, and LSCCi metrics.]

- So inheritance does matter!
  - TCC and LSCC are strongly positively correlated
  - TCCi and LSCCi are strongly negatively correlated

- Several hitherto unknown anomalies exist in these metrics
LSCC prefers the solution on the right, which seems to conflict with OO principles while TCC prevents this refactoring.
Qualitative Analysis

- Looking at *PushDownMethod* more closely yields:

  ```java
  class A {
      void foo() {
          y = 1;
          x = 1
      }
      private int x, y;
  }
  
  class B extends A {
  }
  ```

  ```java
  class A {
      protected int x, y;
  }
  
  class B extends A {
      void foo() {
          y = 1;
          x = 1
      }
  }
  ```

- LSCCi prefers the solution on the left; TCCi prefers that on the right.
CONTRIBUTION

1. Introduction of a novel approach to metric analysis through experimental assessment of software metric using automated refactoring.

2. Propose a quantitative and qualitative insight into similarity and dissimilarity of 5 popular cohesion metrics.

3. In applying this to a set of 5 cohesion metrics, a considerable degree of conflict (38%) was found.

4. Closer examination of two cohesion metrics, TCC and LSCC
   • Including or excluding inheritance has a large impact on a metric
   • Several hitherto unknown anomalies exist in these metrics
Thank You