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.

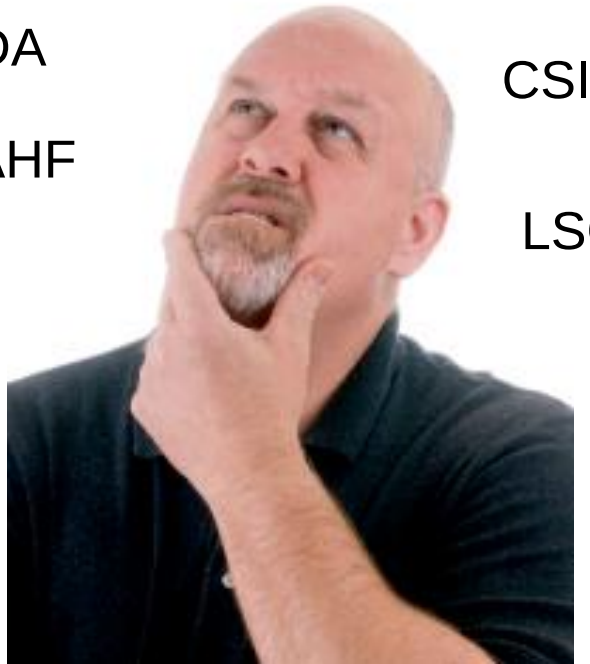
CREST

Roadmap

- Introduction and Motivation
- Experimental Approach
- Code-Imp: our Refactoring Platform
- Experimental Results
- Conclusion

The bewildering world of software metrics

DCC ANA COH DSC CAMC CIS SCOM
ICP CSP
WMC ICBMC DCC CF LCOM4
ICH NOH
DAM LCOM3
MOA RFC DAC CBO
CIDA
NOP CPCC AIF TCC DIT CDP
LCC LSCC
CAM
ICH
LCOM2 COA MIF IIF
CBMC LCOM5
CAI CCDA
CSI
AHEF CCE
NHS CC AHF
MPC
LSCM
LCOM1 NOC CMI
NOM
NHD COF
CIS SNHD
SCC



Analytic approaches have limitations

- Comparing formulae isn't easy:

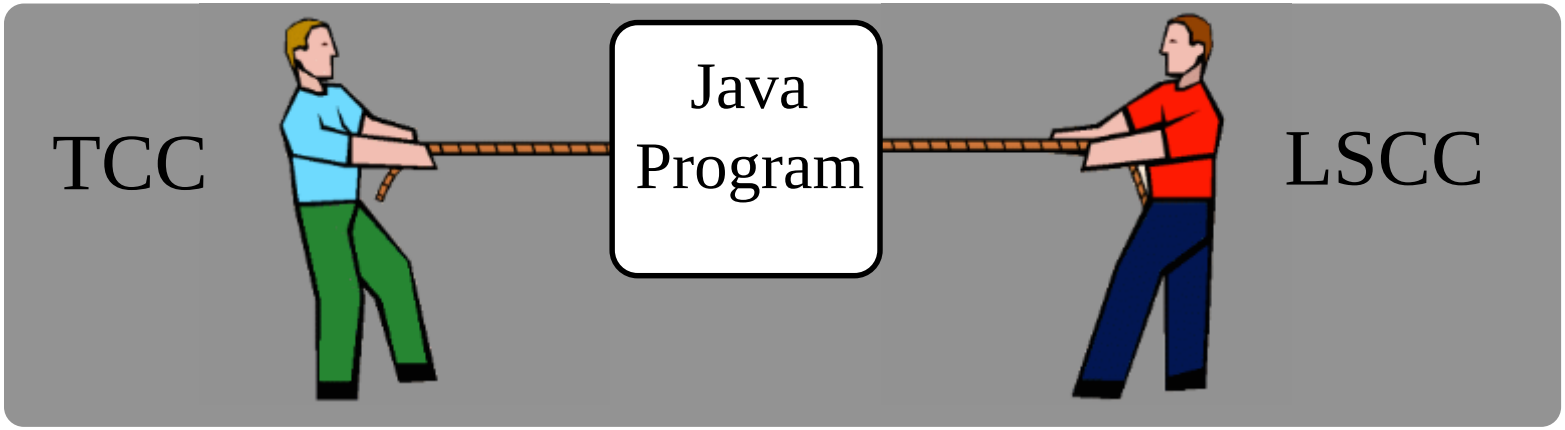
$$\mathrm{CC}(c) = 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{|I_i \cap I_j|}{|I_i \cup I_j|}/k(k-1)$$

$$\mathrm{LCOM5}(c) = \frac{k - \frac{1}{l}\sum_{a \in A_I(c)} |\{m | m \in M_I(c) \wedge a \in \mathrm{I_m}\}|}{k-1}$$

- and may not tell us much about the *practical* aspects of the metric.

Animating the Metrics

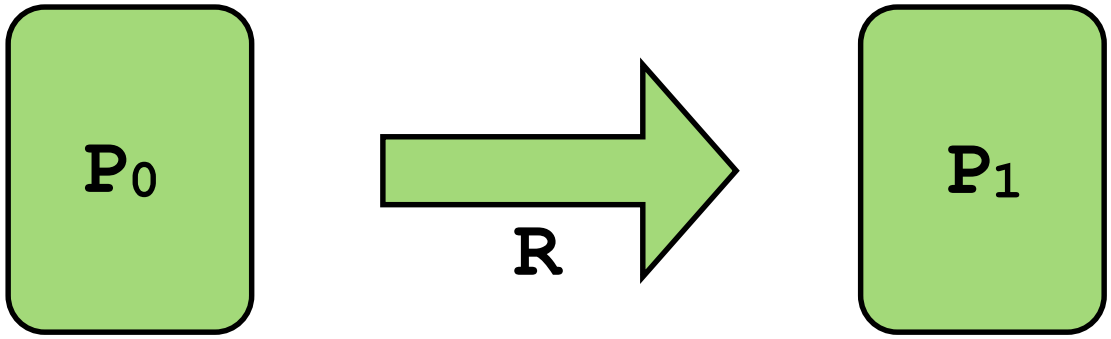
- Our goal is to **animate** the metrics and make then agents of change.



- We use refactorings to change the metrics.

Refactoring and Metrics: an observation

- Refactoring typically has an impact on metrics.



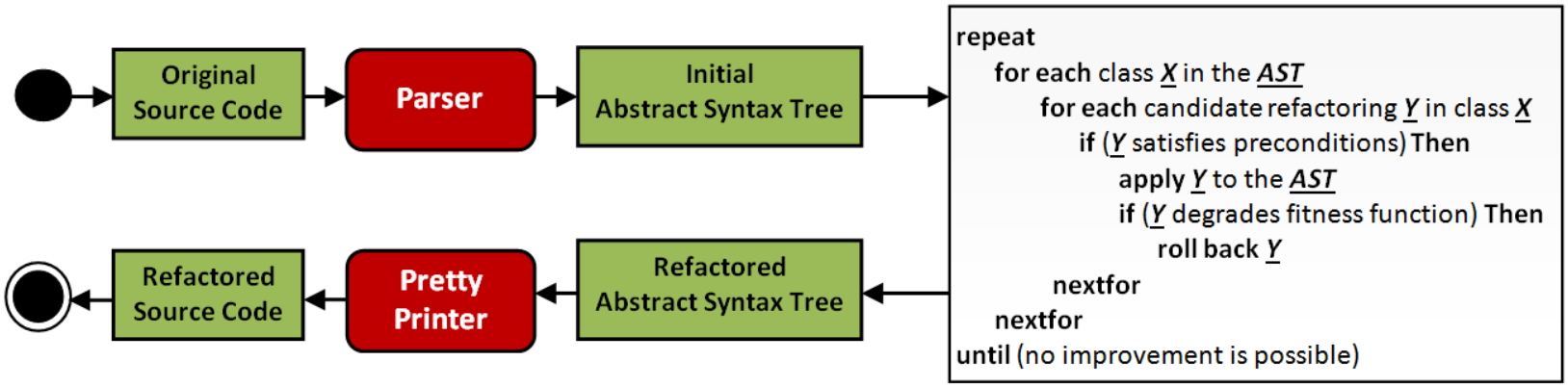
- By calculating metric values before and after applying refactoring R, we observe the behaviour of metrics and learn how they compare with other.

	P_0	P_1
$metric_1$	1.23	1.86
$metric_2$	78.3	62.8

Code-Imp: a Framework for search-based refactoring

Implemented tool: Code-Imp

- An automated search-based refactoring framework



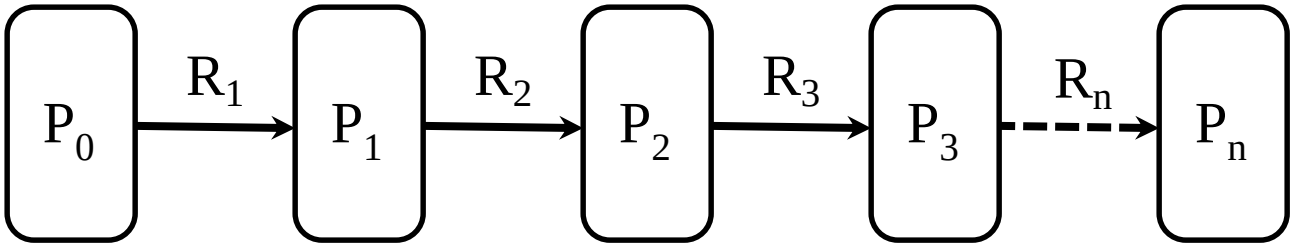
- Three aspects to the refactoring that takes place
  - The set of refactorings that can be applied
  - The type of search technique employed
  - The fitness function that directs the search

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



	P0	**P1**	**P2**	**P3**	.....	**Pn**
metric1	2.12	2.67	2.89	2.50		4.05
metric2	8.73	8.52	8.66	8.88		12.7

Investigation I: General Assessment of Cohesion Metrics

Cohesion Metrics

- In this investigation we explore five popular cohesion metrics:

LSCC	Low-level Similarity-Based Class Cohesion	Al Dallal and Briand, 2010
CC	Class Cohesion	Bonja and Kidanmariam, 2006
SCOM	Sensitive Class Cohesion	Fernández and Peña, 2006
LCOM5	Lack of Cohesion between Methods	Henderson-Sellers, 1996
TCC	Tight Class Cohesion	Biemann and Kang, 1995

Software Analysed

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

Application	# LOC	# Classes
ArtOfIllusion	87,352	459
JabRef	61,966	675
JGraphX	48,810	229
GanttProject	43,913	547
XOM	28, 723	212
JHotDraw	14,577	208
JRDF	12,773	206
JTar	9,010	59

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

Experiment and Results

Metric volatility as a percentage

	JHotDraw (1007)	JTar (115)	XOM (193)	JRDF (13)	JabRef (257)	JGraph (525)	ArtOfIllusion (593)	Gantt (750)	All (3453)
LSCC	96	99	100	92	99	100	99	96	98
TCC	86	53	97	46	61	72	84	71	78
SCOM	79	70	93	92	79	89	77	80	81
CC	100	98	100	92	99	100	100	99	100
LCOM5	100	100	100	100	100	100	100	99	100

- 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

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

Spearman rank correlation between the metrics

	LSCC	TCC	SCOM	CC
TCC	0.60			
SCOM	0.70	0.58		
CC	0.10	0.01	-0.28	
LCOM5	-0.17	-0.21	-0.46	0.72

Metric Conflict

- We categorise each metric pair as follows:

Agreement	Both metrics improve, disimprove, or remain the same
Dissonance	One metric changes while the other remains the same
Conflicted	One metric improves while the other disimproves

- 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**.

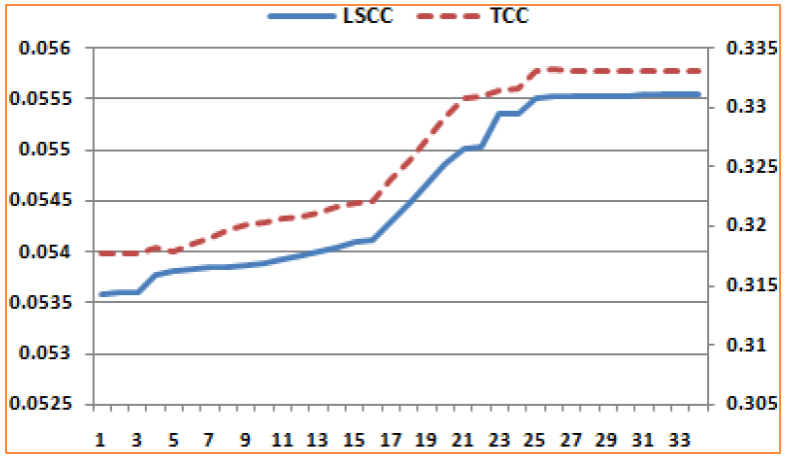
Fitness Function

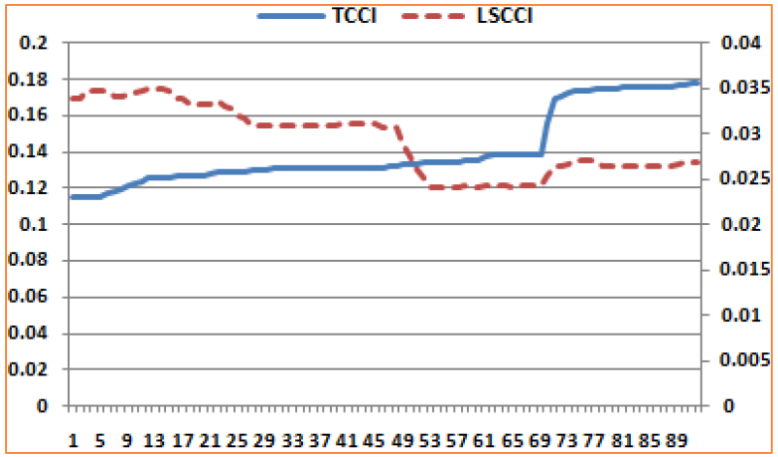
- A refactoring is accepted only if it is Pareto optimal across all the classes of the application.

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

Experiments and Results

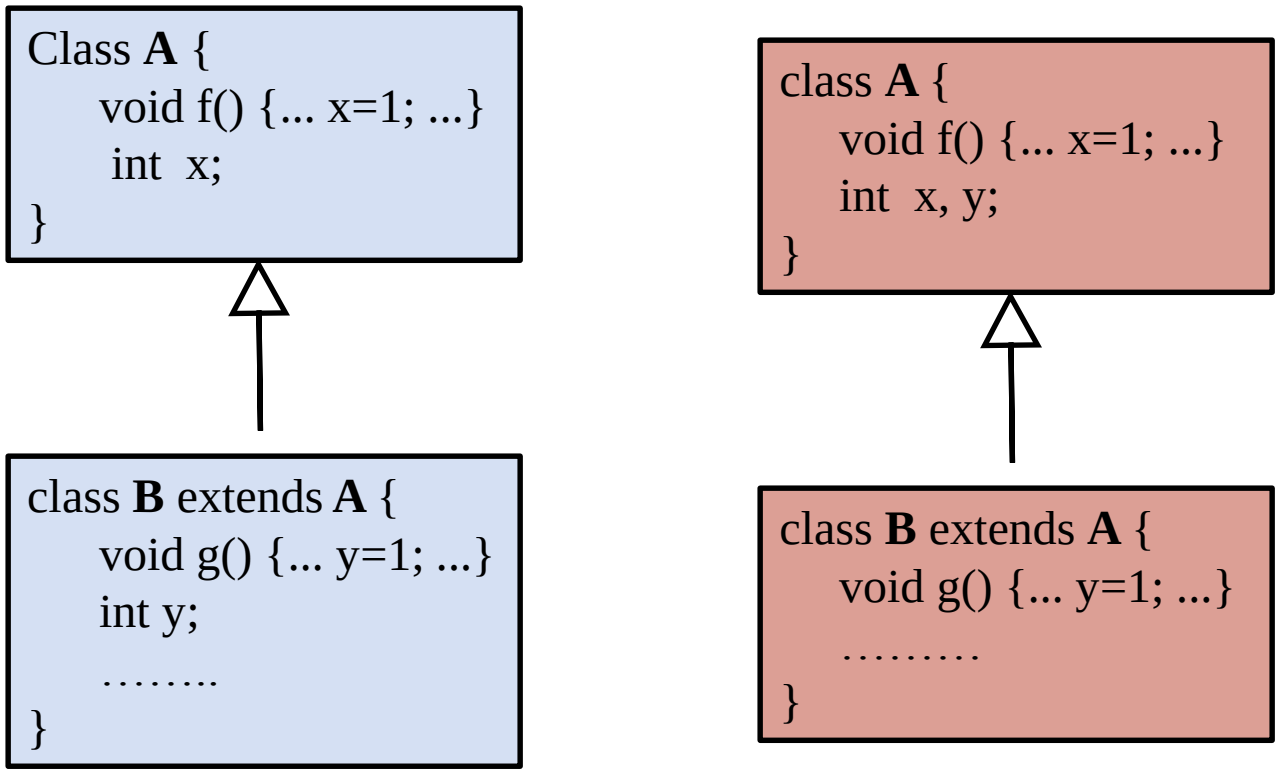
- To inherit or not to inherit





- 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

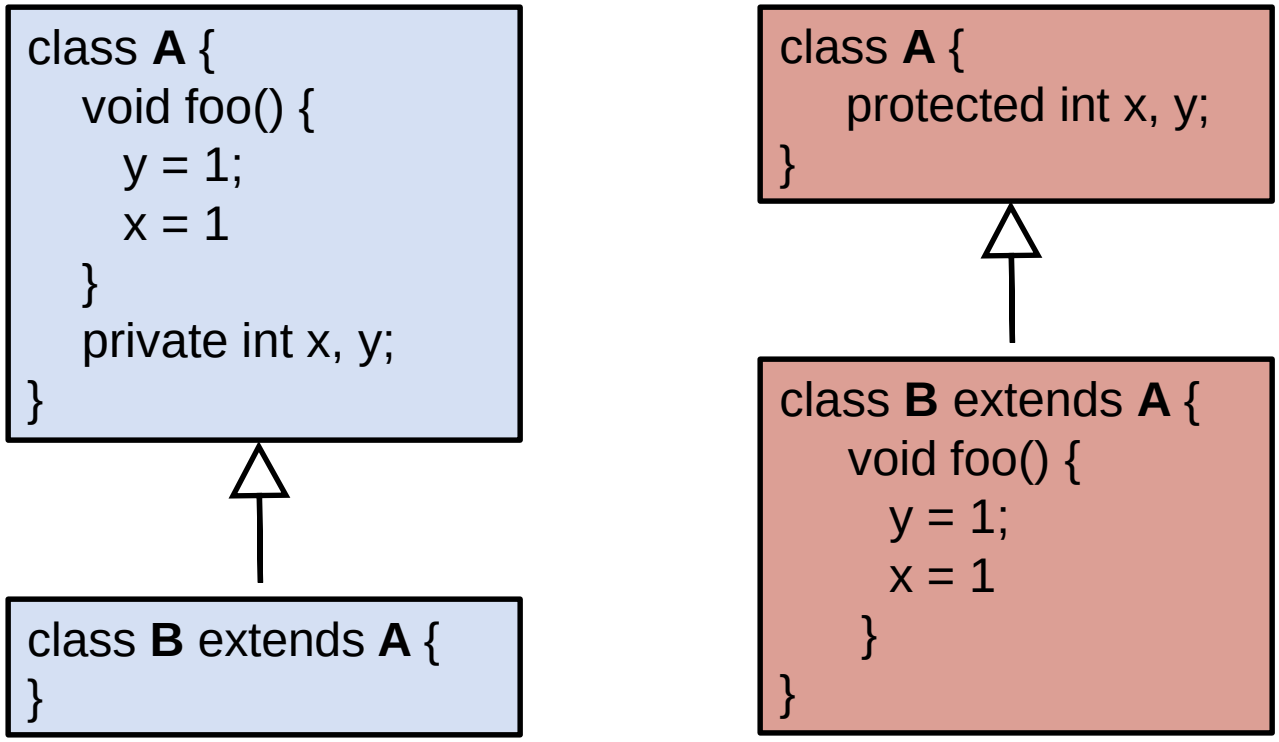
Qualitative Analysis



- 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:



- LSCC$_i$ prefers the solution on the left; TCC$_i$ 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