Software Improvement Group

Static Estimation of Test Coverage

Tiago Alves & Joost Visser

September 20, 2009

Arent Janszoon Ernststraat 595-H
NL-1082 LD Amsterdam
info@sig.nl
www.sig.nl

Motivation

Software Improvement Group

Test Coverage in Software Quality assessment:

- NO full installation available (missing sources or libraries)
- NO execution (unavailable hardware, time constraints, no reproducible test environment)
  - NO instrumentation of source/byte code

Research questions:

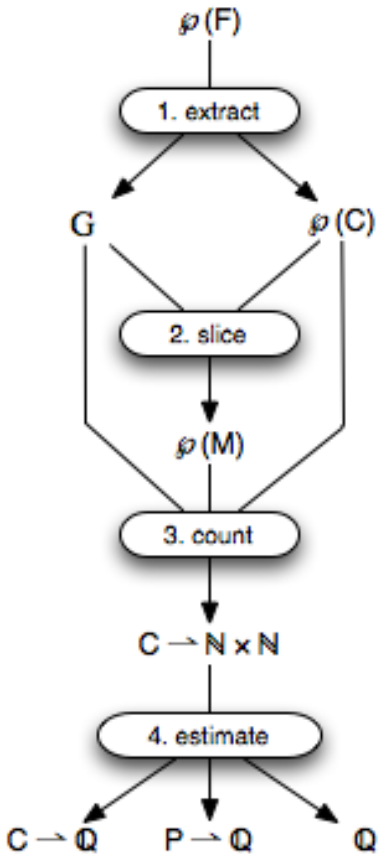
- Is it possible to determine test coverage without running tests?
- What trade-offs can be made between scalability and precision?

Requirements

- Use only static analysis
- Scale to large systems
- Robust against incomplete systems

Solution sketch

Software Improvement Group



Static estimation of test coverage

1. **Extract**
   - Extract structural and call information
   - Determine set of test classes
2. **Slice (modified)**
   - Slice graph starting from the test methods
   - Set of methods reached from test code
   - Take into account class initializer calls
3. **Count (per class)**
   - Determine number of defined methods
   - Determine number of covered methods
4. **Estimate method coverage**
   - Class level
   - Package level
   - System level

Experimental design

Software Improvement Group

Data set selection and characterization

- 12 Open-source and proprietary Java systems (1.6k - 267k LOC)

Execution of experiment

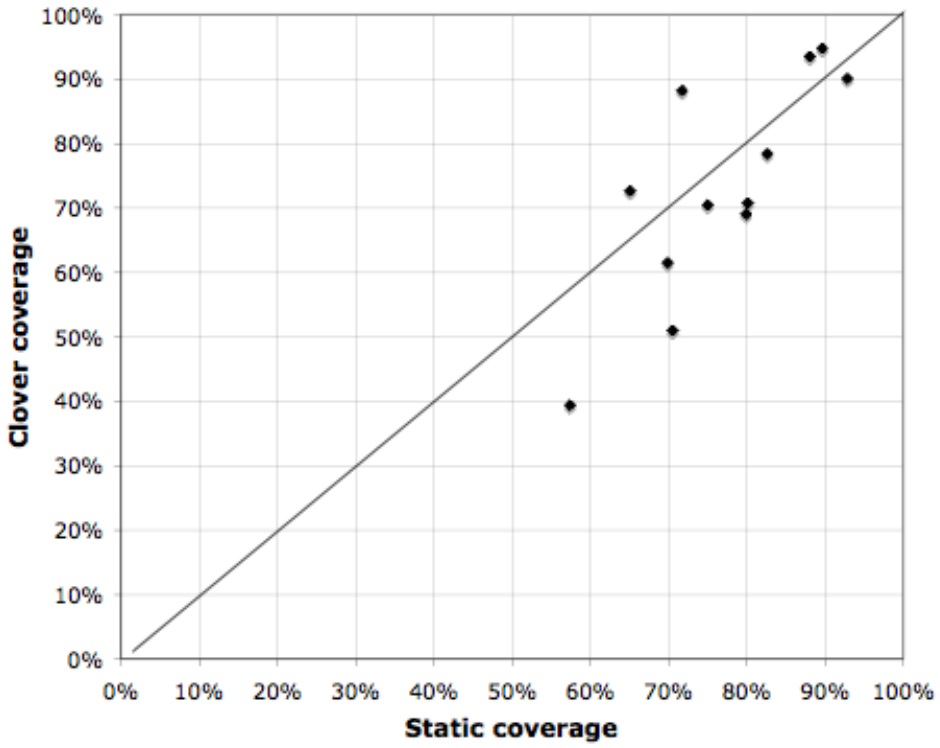
- SemmleCode execution (text file export + scripts for CSV conversion)
- XML Clover extraction (XSLT transformations to CSV conversion)
- Custom built java tool to read CSV files and XLS creation

Statistical analysis

- Histograms (distribution)
- Scatter charts (correlation)
- Spearman (correlation)
- Inter-quartile ranges (dispersion)

Statistical analysis
(System coverage comparison)

Software Improvement Group

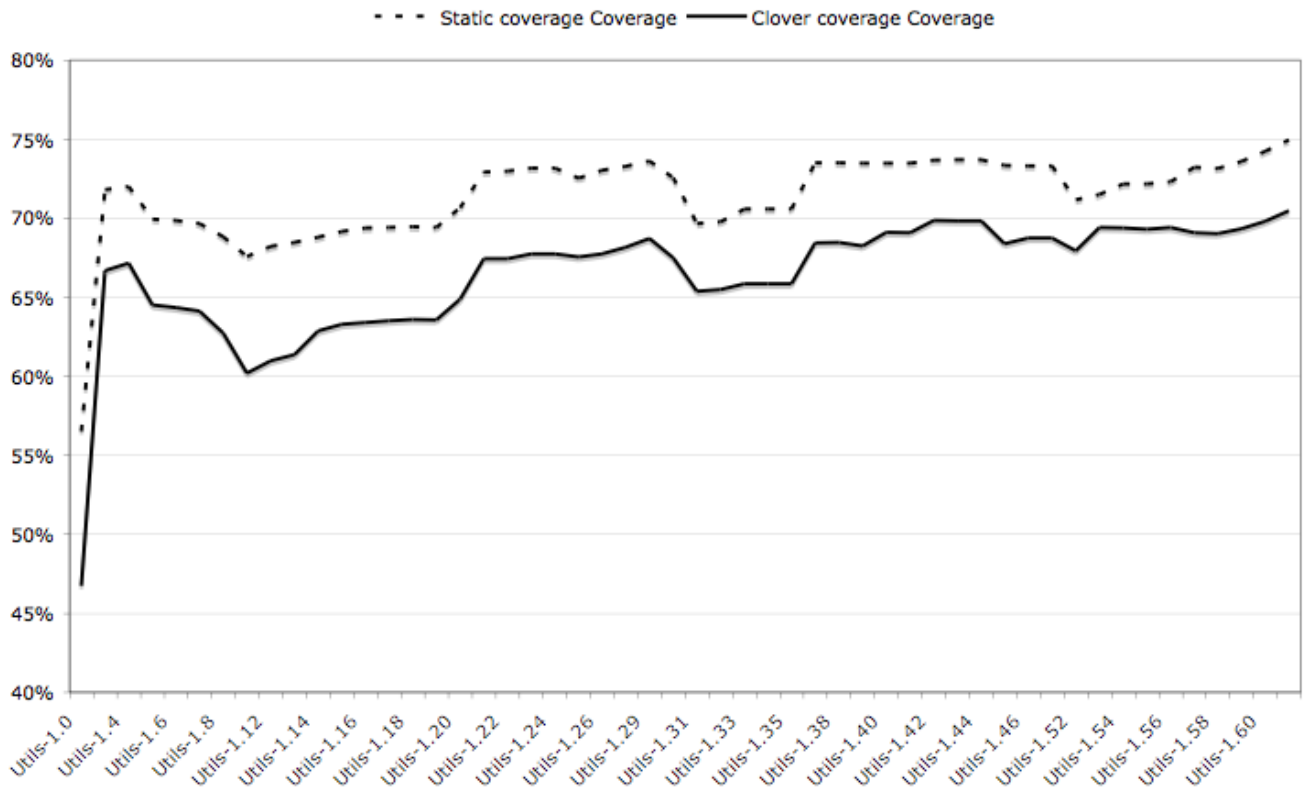


Spearman correlation: 0.802**

System	Static	Clover	Diff
JPacMan	88.06%	93.53%	-5.47%
SIG Certification	92.82%	90.09%	2.73%
G System	89.61%	94.81%	-5.19%
Dom4j	57.40%	39.37%	18.03%
SIG Utils	74.95%	70.47%	4.48%
JGAP	70.51%	50.99%	19.52%
Collections	82.62%	78.39%	4.23%
PMD	80.10%	70.76%	9.34%
R System	65.10%	72.65%	-7.55%
JFreeChart	69.88%	61.55%	8.33%
SIG DocGen	79.92%	69.08%	10.84%
SIG Analyses	71.74%	88.23%	-16.49%

Statistical analysis
(System coverage through time)

Software Improvement Group



Spearman correlation: 0.888**

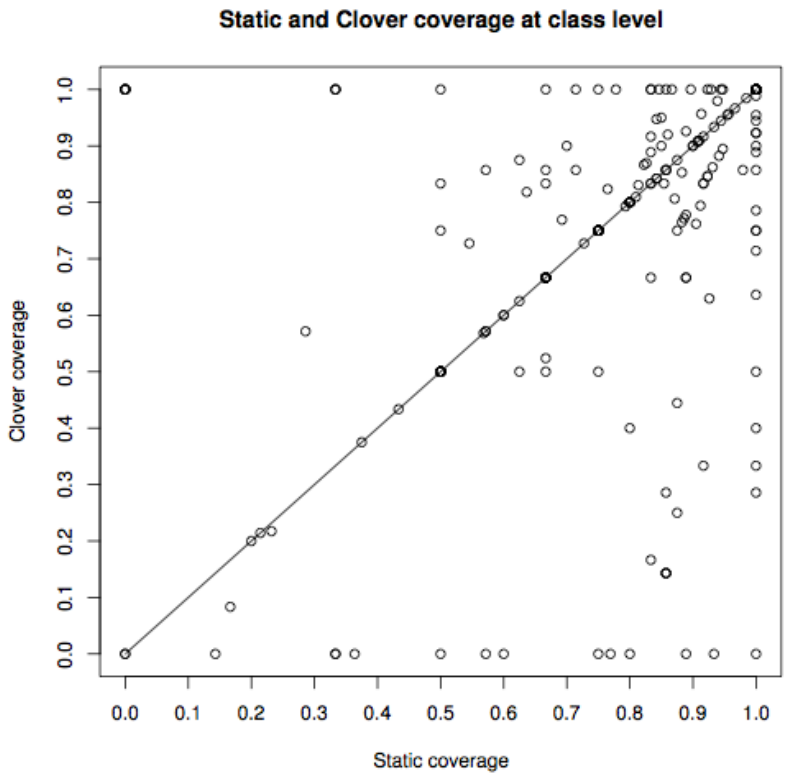
Statistical analysis
(Class and package coverage comparison)

Software Improvement Group

System name	Spearman	
	Class	Package
JPacMan	0.476*	1
SIG Certification	0.368**	0.520
G System	0.774**	0.694**
Dom4j	0.584**	0.620*
SIG Utils	0.825**	0.778**
JGAP	0.733**	0.786**
Collections	0.549**	0.776**
PMD	0.638**	0.655**
R System	0.727**	0.723**
JFreeChart	0.632**	0.694**
SIG DocGen	0.397**	0.459**
SIG Analyses	0.391**	0.486**

Static estimation of test coverage

Collections



Why the values differ? (Sources of imprecision)

Software Improvement Group

<table>
<tr><th>Overestimation</th><th>Underestimation</th></tr>
<tr><td>Control flow</td><td rowspan="4">Frameworks / Library call backs</td></tr>
<tr><td>Dynamic dispatch</td></tr>
<tr><td>Overloading</td></tr>
<tr><td>Failing tests
(when comparing w/ Clover)</td></tr>
<tr><td colspan="2">Identification of test cases</td></tr>
</table>

Conclusion

Software Improvement Group

It is possible to determine test coverage without running tests!

- Spearman: high correlation between static and clover coverage
- In general static coverage identifies the same values as clover

Trade-offs between scalability and precision?

- Average absolute difference for system coverage: 9%
- Class and Package coverage needs further improvement

Future research

- Use LOC as weight for better estimation
- Merge with Kastrén “Towards a deeper understanding of test coverage”
- More heuristics: McCabe , #Tests, #asserts, Test(LOC) / Code(LOC)

Questions?