Investigating the Impact of Design Debt on Software Quality

Prioritizing Design Debt Investment Opportunities

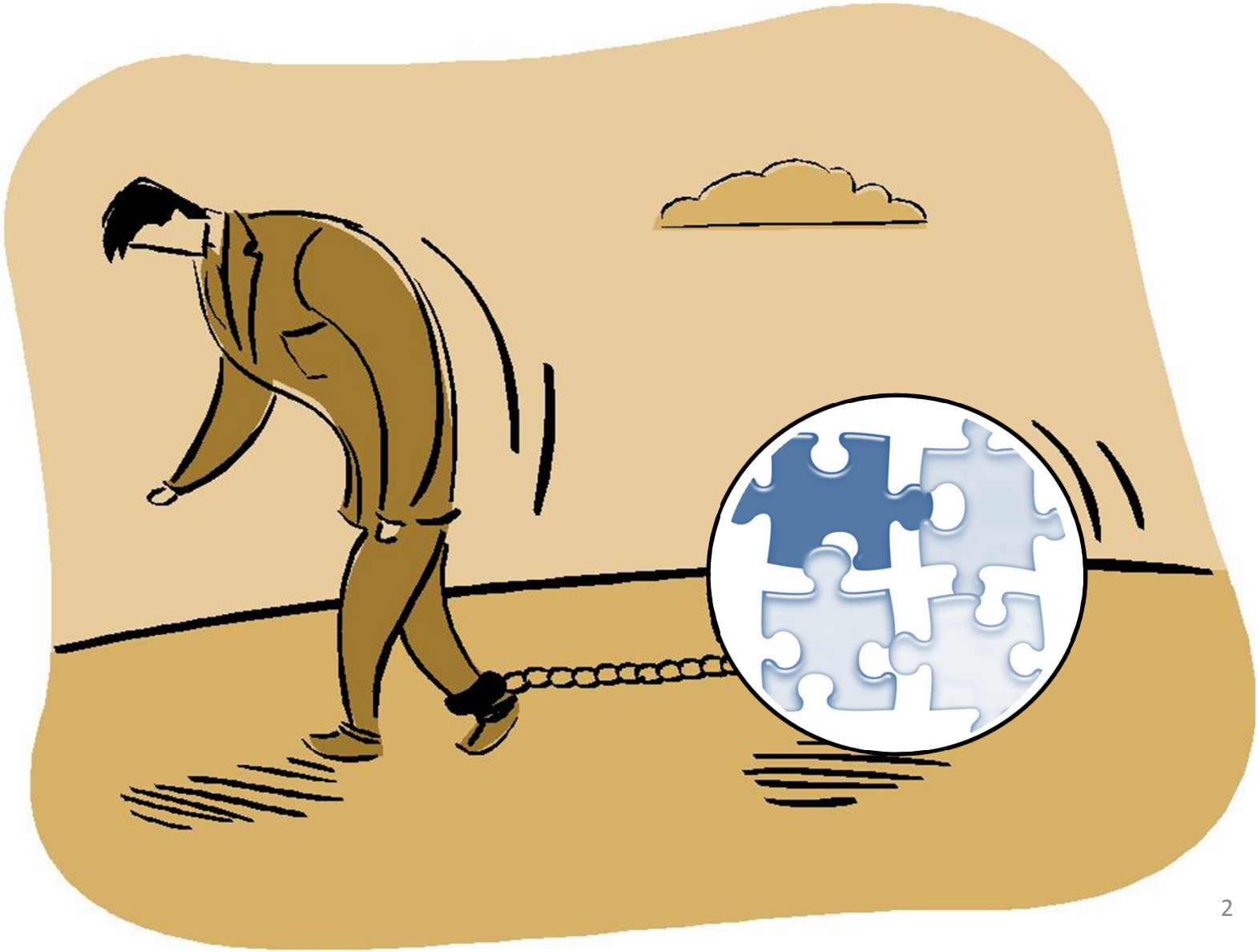
Nico Zazworka

Carolyn Seaman, Forrest Shull, Michele Shaw

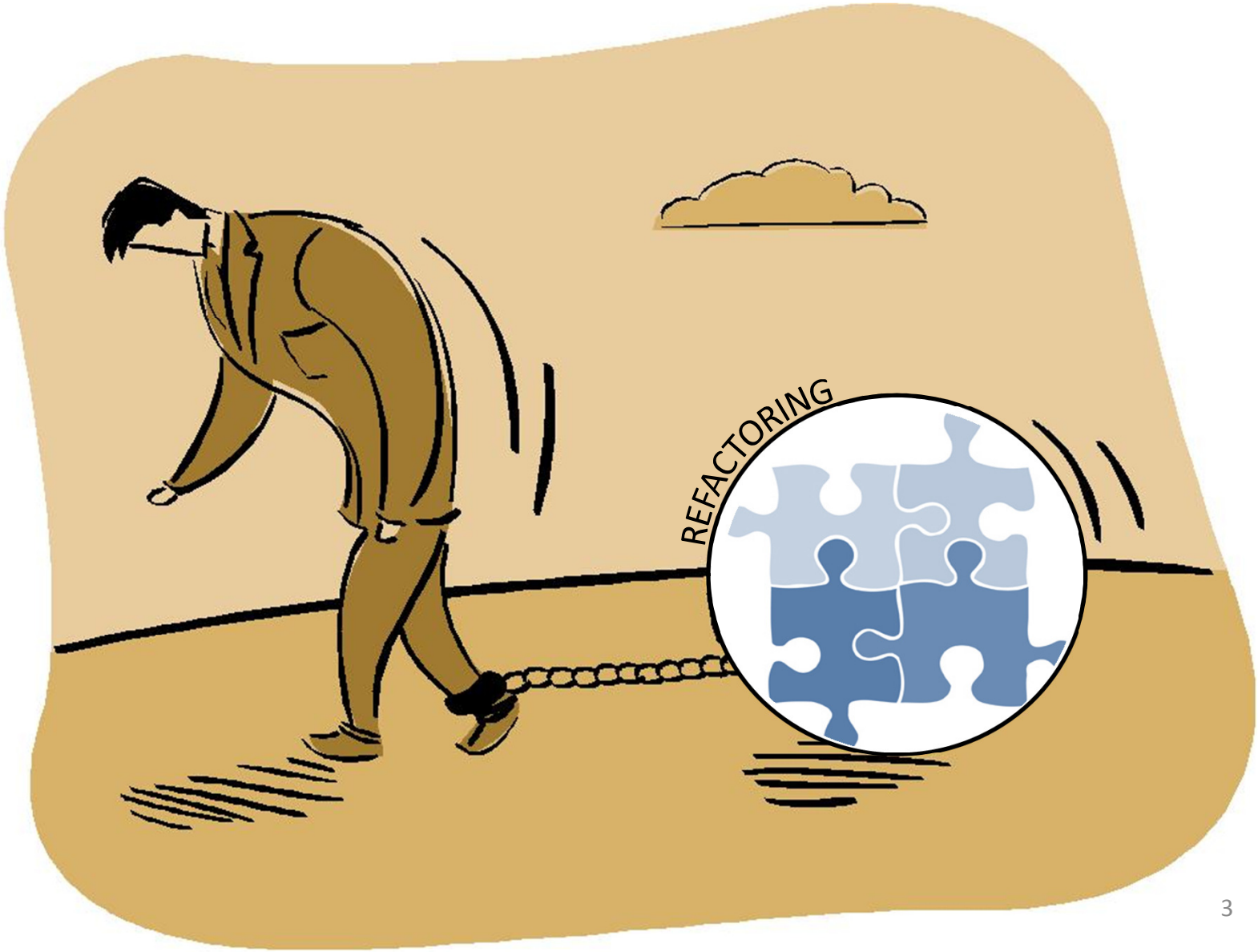
Fraunhofer USA
Center for Experimental Software Engineering

UMBC
AN HONORS UNIVERSITY IN MARYLAND

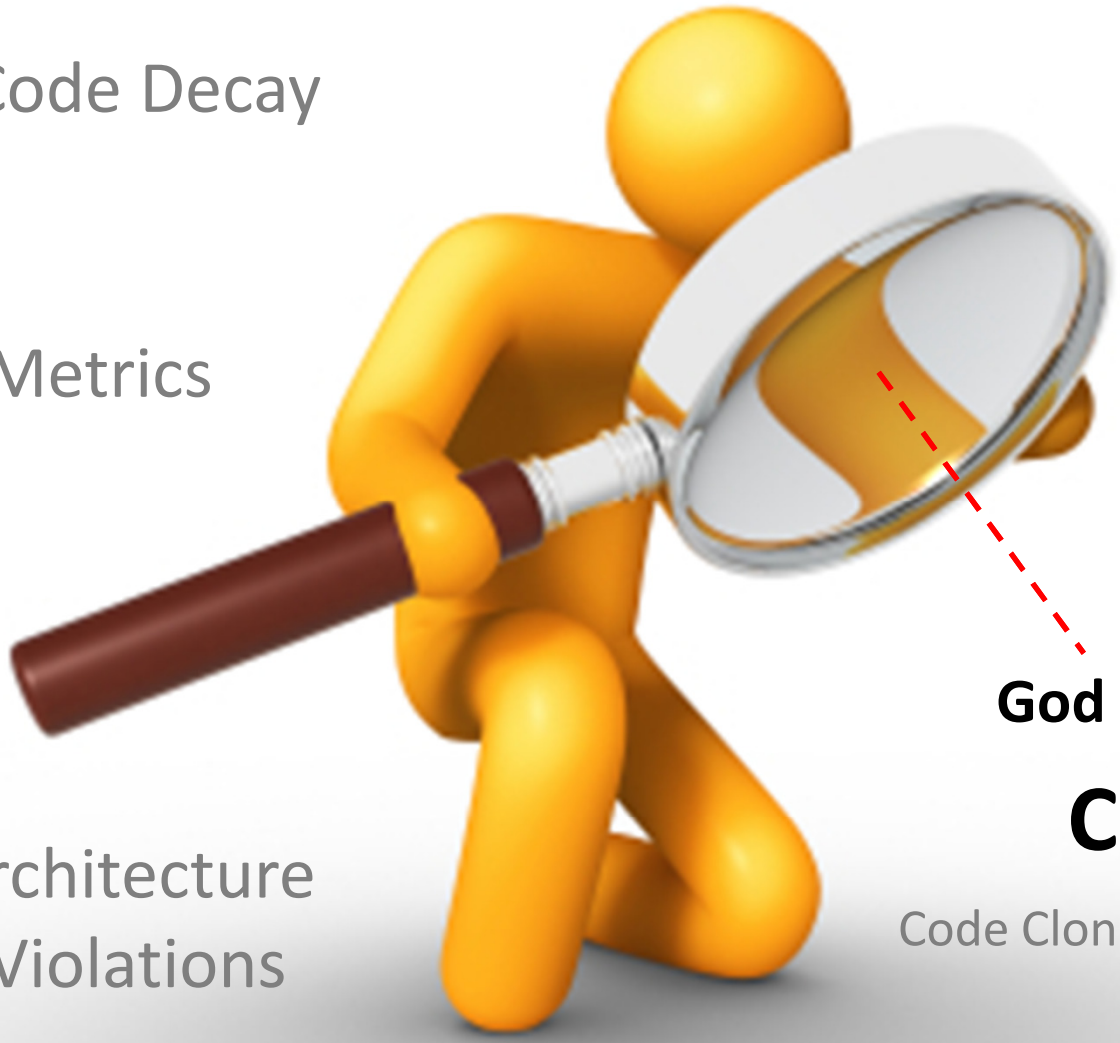
Design Debt



Design Debt



Potential Indicators



Code Decay

Lack of
Design Patterns

Code Metrics

God Class

Data Class

Code Smells

Architecture
Violations

Code Clones

Tradition Breaker

Intensive Coupling

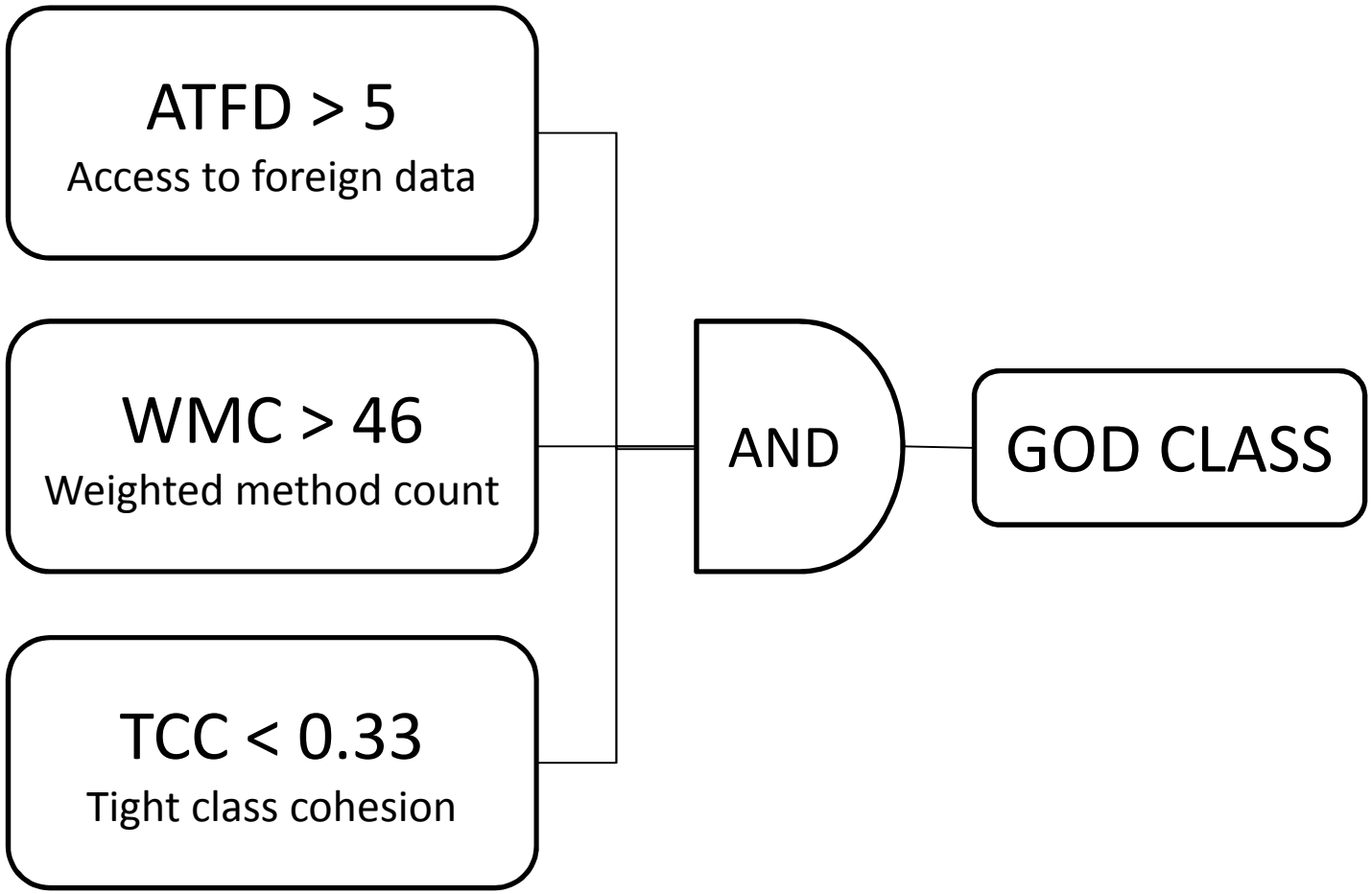
Research Questions

- Are Code Smells, i.e. God Classes, valid indicators for design debt?
  - Do God Classes have a negative impact on:
    - Maintainability and
    - Correctness
- Can we give advice on which design debt to pay first?
  - Which God Classes are easy to fix and promise high gain in software quality?
  - Which God Classes are hard to fix and promise low gain in software quality?

The God Class

- Also known as “Large Class” [Fowler99]
- Marinescu [Mar04]
  - Centralizes intelligence
  - Multiple responsibilities
  - Delegates minor detail
  - Uses data of other classes

God Class Detection



Case Study

- Small software development company
  - 30 employees: C# developers, web-designers
  - 2 active development projects
    - Project J: 35kLOC, 11 months, 4 developers
    - Project F: 45kLOC, 17 months, 4 developers
- Previously performed a code smell study in the same environment
- Small part of developers were familiar with technical debt metaphor
- Data: subversion repository and JIRA bug tracker

God Classes and Maintainability

- Assumption: maintainability can be estimated by investigating how often a class to be changed
  - Rational: classes that have to be changes too often, e.g. with each revision, are indicators for maintenance bottlenecks
- H1: The change likelihood of god classes is higher than for non-god classes

Revision	1452	1457	1471	1472	1424	Likelihood
Changed God Classes	0/4	1/4	1/4	2/4	2/4	0.300
Changed Non-God Classes	1/223	4/223	6/225	4/225	2/225	0.015

Example for change likelihood for god classes and non-god classes in project F

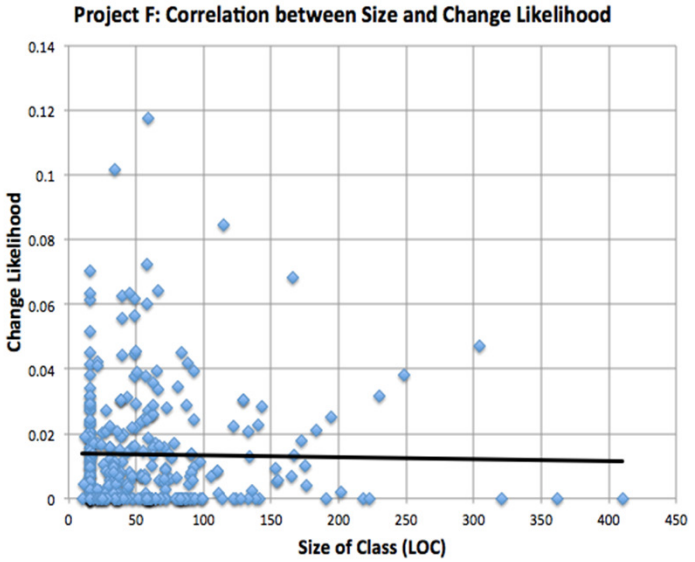
Maintainability Results

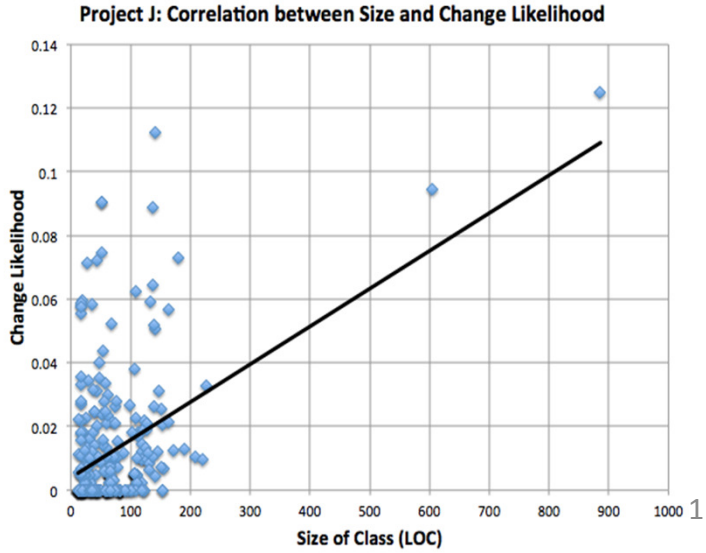
	Project F		Project J	
	God Classes	Non-God Classes	God Classes	Non-God Classes
N	545	658	282	328
mean	0.07848	0.01619	0.12565	0.01725
s	0.18448	0.03837	0.24754	0.02391
	p-value: 4.282e-14		p-value: 2.461e-12	

- God classes are 5-7 times more change prone
- Do we need to normalize this data by size?

Investigating Normalization

- Assumption: “A class that is twice as large, is twice as change prone.”
- Method: Measure correlation between:
  - Size (LOC)
  - Change Likelihood
- Results (Pearson CC):
  - Project F: **-0.029**
  - Project J: 0.42
- Dividing by LOC might over-normalize result
  - Project J normalized result still statistically significant





God Classes and Defects

- H2: The defect likelihood of god classes is higher than for non-god classes
- Data: JIRA bugs are linked to subversion change sets (=classes that were part of the bug fix)

Defect (JIRA issue)	J-166	J-161	J-377	J-396	J-228	Likelihood
Fix Revisions	9097, 9098	8939	11990	12842, 12844	10269	
God Classes	1/3	0/1	0/8	3/8	0/3	0.1417
Non-God Classes	0/94	1/94	1/156	0/157	1/101	0.0067

Example for defect fix likelihood for god classes and non-god classes in project J

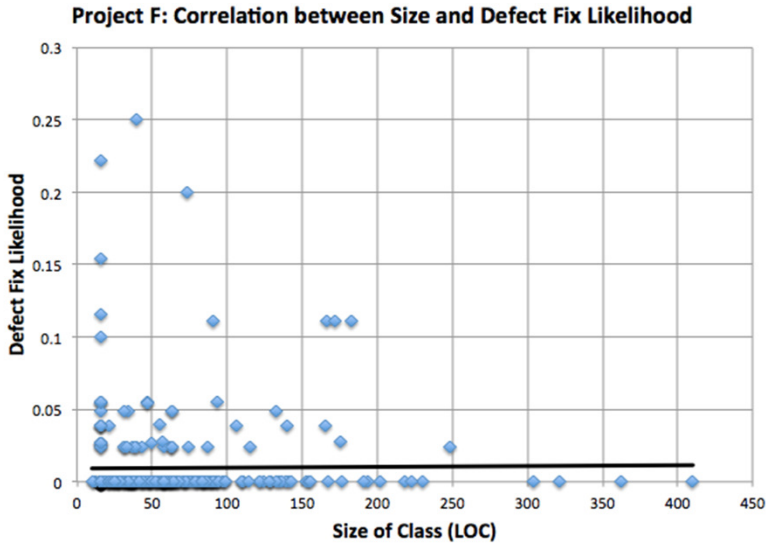
Defect Results

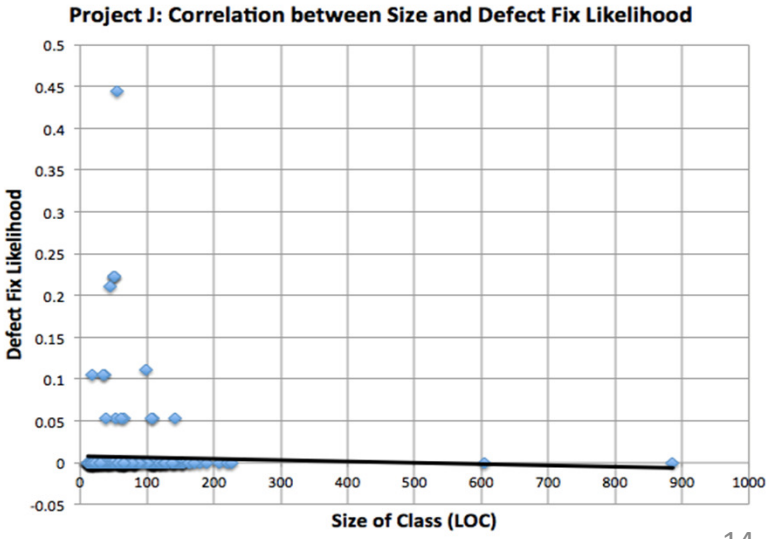
	Project F		Project J	
	God Classes	Non-God Classes	God Classes	Non-God Classes
N	32	32	17	17
mean	0.03939	0.00956	0.16911	0.00624
s	0.13669	0.01094	0.22266	0.00796
	p-value: 0.2276 (not sig.)		p-value: 0.008217	

- God classes are 4-17 times more defect prone
- Do we need to normalize this data by size?

Investigating Normalization

- Assumption: “A class that is twice as large, is twice as defect prone.”
- Method: Measure correlation between:
  - Size (LOC)
  - Defect Likelihood
- Results (Pearson CC):
  - Project F: 0.011
  - Project J: -0.018
- Dividing by LOC will over-normalize result





Related Research

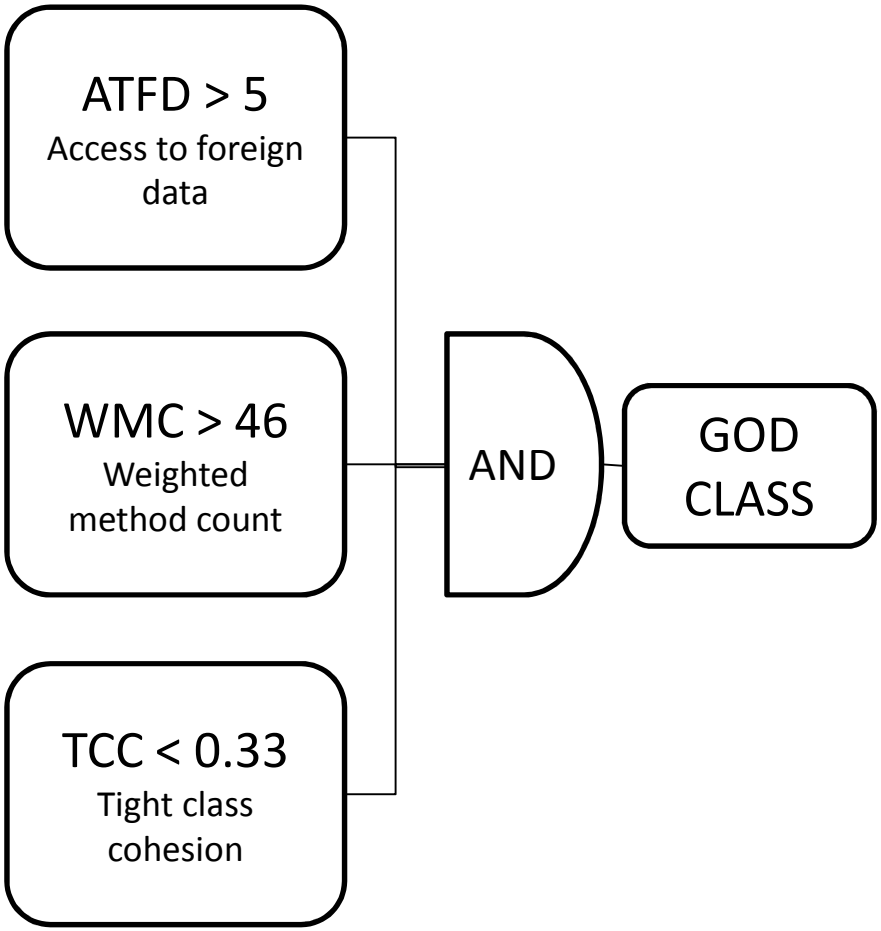
Related Work	Investigated Software	God classes more change prone if not normalized? (p<0.05)	God classes more change prone if LOC normalized? (p<0.05)	God classes more defect prone if not normalized? (p<0.05)	God classes more defect prone if LOC normalized? (p<0.05)
Li 2007	Eclipse				
Olbrich 2009	Lucene, Xerces				
Schumacher 2010	Two commercial applications				
Olbrich 2010	Lucene, Xerces, Log4j		 less change prone	 in 2 out of 3 cases	 less defect prone in 2 out of 3 cases
Khomh 2009	Azereus, Eclipse	 5 out of 10 releases			
Study results presented here	Two commercial applications		 in 1 out of 2 cases	 1 out of 2 cases	 in 1 out of 2 cases

Paying Design Debt

- Moving from identifying TD to **managing TD**
- Paying off debt is an **investment opportunity** with tradeoffs:
  - Value of debt (how much is it going to cost to fix it?)
  - Interest rate (how much does it slow down development?)
  - Probability (what is the chance that the debt affects productivity?)
- Goal: select the most profitable opportunities, ignore non-profitable ones.
- Profitable (good cost/benefit ratio)
  - Low value
  - High interest rate

Cost of Paying Debt

- Refactoring
- Idea: facilitate metrics in detection model
- Argument: a class being close to the thresholds will be easier to refactor than one that is multiple magnitudes outside.
- Method: rank god classes according to their distance to the thresholds



God Class Ranking: Cost

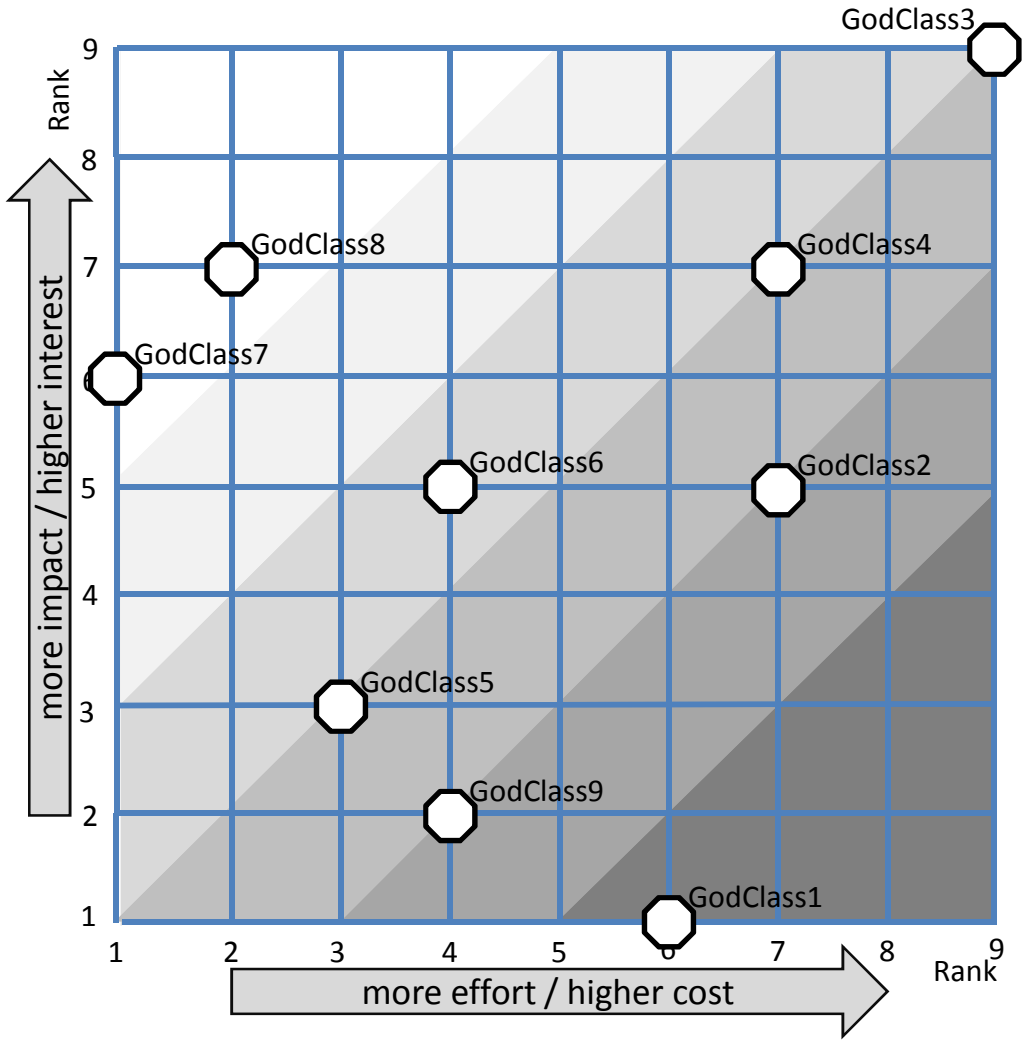
God Class Name	WMC (>46)		TCC (<0.33)		ATFD (>5)		Overall Score and Rank	
	Value	Rank	Value	Rank	Value	Rank	Sum Rank	Rank
GodClass1	49	3	0.0	8	20	6	17	6
GodClass2	87	8	0.005	7	28	7	22	7
GodClass3	107	9	0.0	8	28	7	24	9
GodClass4	69	7	0.026	6	34	9	22	7
GodClass5	49	3	0.065	5	9	3	11	3
GodClass6	60	5	0.177	4	19	4	13	4
GodClass7	47	1	0.219	1	7	1	3	1
GodClass8	48	2	0.199	2	7	1	5	2
GodClass9	61	6	0.192	3	19	4	13	4

God Class Ranking: Interest

- Interest: negative effect on software quality
  - Maintainability
  - Defects
- Method: use change and defect likelihood to estimate and rank impact

God Class Name	Change Likelihood		Defect Likelihood		Overall Score and Rank	
	Value	Rank	Value	Rank	Sum Rank	Rank
GodClass1	0.016	1	0.0	1	2	1
GodClass2	0.097	8	0.0	1	9	4
GodClass3	0.102	9	0.029	5	14	9
GodClass4	0.068	7	0.177	6	13	7
GodClass5	0.040	3	0.0	1	4	3
GodClass6	0.0455	4	0.133	7	11	5
GodClass7	0.0458	5	0.133	7	12	6
GodClass8	0.052	6	0.133	7	13	7
GodClass9	0.027	2	0.0	1	3	2

Cost/Benefit Matrix



Future Work

- Evaluation of other code smells and other indictors
- Empirical evaluation of cost/benefit model
  - Are our assumptions on correlation of metrics and refactoring cost true?
  - Are god classes after refactoring indeed less change and defect prone?
  - Can we advance from a ranking to a more precise prediction model?
- Managing design debt and god classes:
  - When should a god class be refactored?
  - When is it acceptable to introduce a god class for short term gains?

QUESTIONS?

Fraunhofer USA

Dr. Nico Zazworka
Research Scientist

Center for Experimental Software Engineering
University of Maryland
Phone: 240 487 2928
Email: nzazworka@fc-md.umd.edu