


Test Coverage and Post-Verification Defects: A Multiple Case Study

A. Mockus - audris@avaya.com

N. Nagappan - nachin@microsoft.com

T. Dinh-Trong - ttdinhtrong@avaya.com



*Avaya Labs Research
Basking Ridge, NJ 07920
<http://mockus.org/>*

Outline

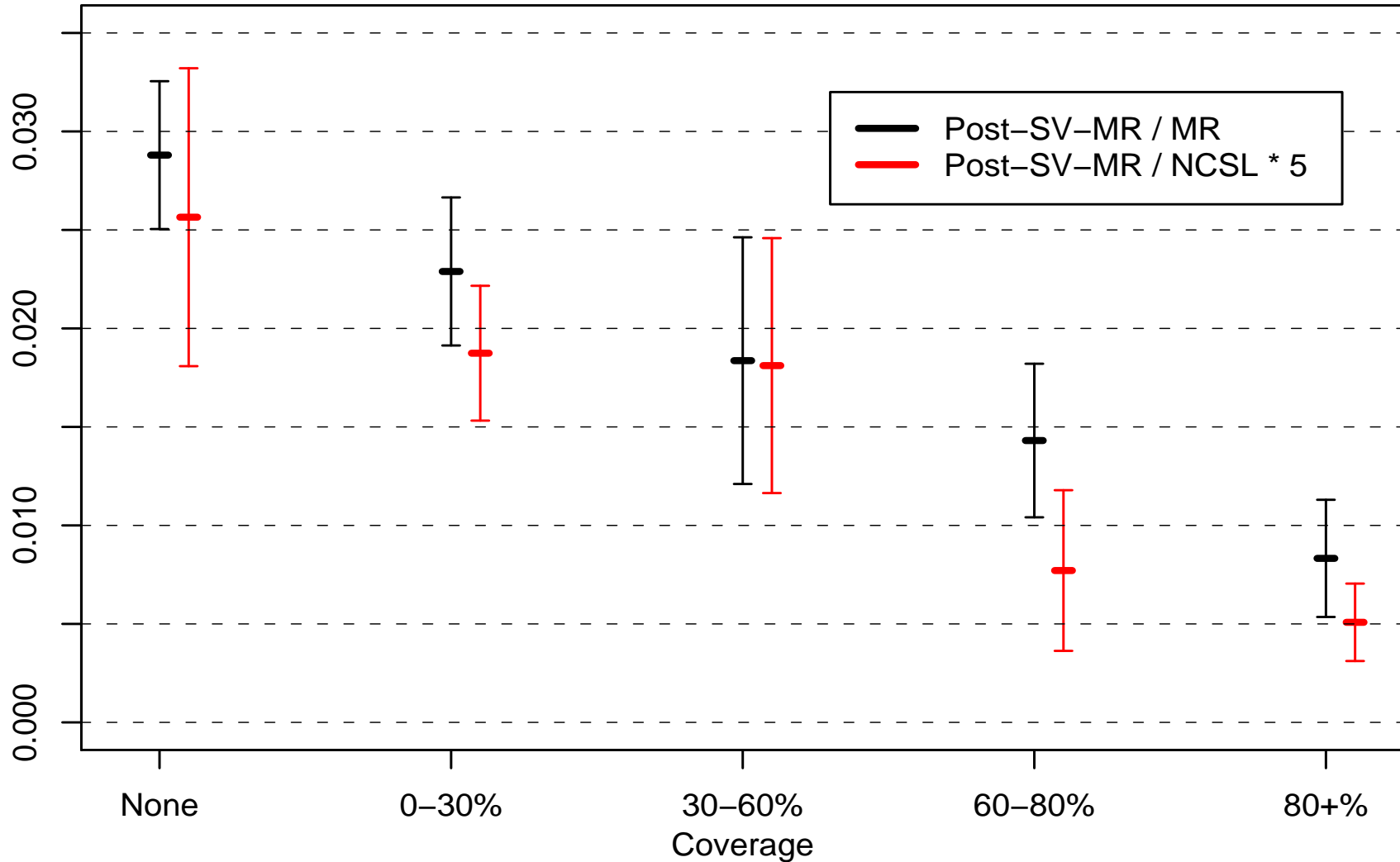
- ❖ Background and motivation
- ❖ Method and context
- ❖ Qualitative evidence
- ❖ Coverage and defects
- ❖ Coverage and effort
- ❖ Validity
- ❖ Summary

How much code coverage is needed?

❖ Hypotheses

- ❖ Increasing the level of coverage would decrease the defect rate.
- ❖ Progressively more effort is needed to increase the coverage by the same amount for higher levels of coverage.

Increasing coverage decreases defect rate



Context: projects A and M

Area	Avaya	Microsoft
Domain	Real-Time Reporting	OS
Language	Java	C/C++
Size	1 MLOC	40+ MLOC
Domain	Application	System
Team size	≈ 100	1000+
Users	≈ 1000	$\approx 100M$
Testing phase	Unit	System
Tools	CC/CQ/JUnit	In-house
Types of coverage	Statement, branch	Block, Arc
Granularity	Files	Binaries

Method

- ❖ Literature review
- ❖ Developer interviews
- ❖ Coverage data
- ❖ Source code change data
- ❖ Test creation
- ❖ Post-analysis interviews

Qualitative Method

❖ Aim

- ❖ To understand the possible mechanisms of the relationship between code coverage and customer-reported defects

❖ Method

- ❖ Microsoft: Informal interviews with senior engineers (> 10 Years) and senior engineers in teams with high coverage to more failures
- ❖ Avaya: structured interviews with senior and junior developers and managers. Also used to validate defect, change, effort, and coverage data

Qualitative results: Microsoft

- ❖ Covered \neq correct: if the testing scenario was wrong and not representative we may have high coverage but low relationship to field usage.
- ❖ Usage is important: if 90% of code is covered and never used in the field or conversely if the non-covered 10% is the only part of the codebase used extensively in the field then it is more likely to have defects due to high usage
- ❖ Complexity is important : coverage of 80% on a file with CC 1000 is much harder than for a file with CC 10

Qualitative results: Avaya

- ❖ “When I develop files from scratch, I tend to write more test cases for it. If I just maintain the files, I would rarely write test cases for it.”
- ❖ “The files that are complex and are used by many other files tend to be tested more.”
- ❖ “The files that are easier to test tend to be tested more. When we first deployed code coverage measurement practice, we first write test cases for these files to quickly increase the overall coverage.”
- ❖ “The files that provide a service class tend to be called by many different classes. Hence it is executed more during testing even though we might not intend to test them.”
- ❖ “Some areas, such as UI, are harder to test and hence tend to be tested less than the other.”
- ❖ “Code that interacts with database might not be well-tested because it may take a lot of effort to write stubs representing database behavior.”
- ❖ “Remote development locations may not be using test coverage as extensively.”

Quantitative data

- ❖ Coverage
 - ❖ Avaya: Cobertura - statements and branches
 - ❖ Microsoft: Internal tool - block and arc coverage
- ❖ Changes and Defects: link to coverage at file/class level
 - ❖ Avaya: ClearCase and ClearQuest
 - ❖ Microsoft: Internal tools
- ❖ Testing effort (JUnit)
 - ❖ Modifications to test classes corresponding to the class being tested

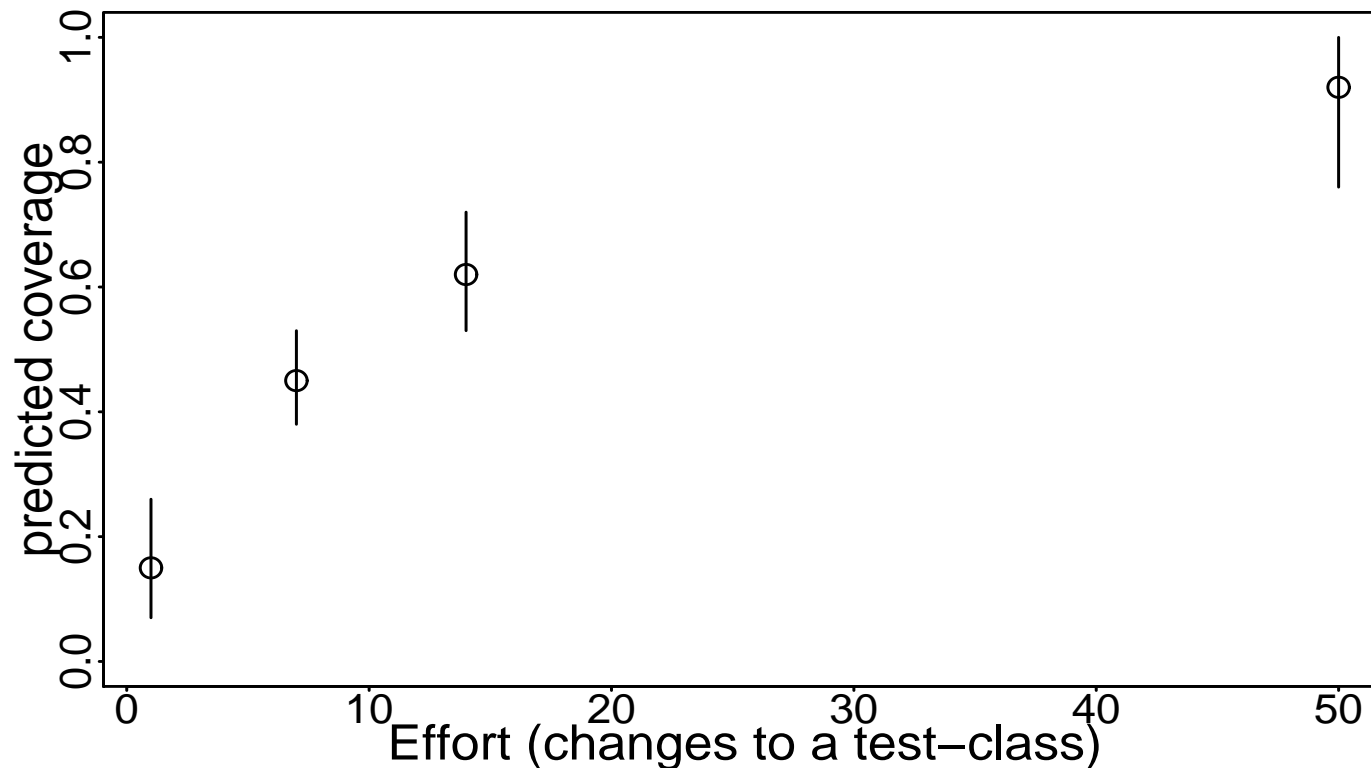
Increasing coverage decreases defect rate

- ❖ Response: number of customer reported failures
- ❖ Observations: M: NA, A: 2472
- ❖ Fit: M: $R^2 = 0.66$, A: 20% of the deviance explained

	M:t-val	M:p-val	A: Est.	A: Stderr	A:p-val
(Intercept)	12.3	<0.0005	-4.72	0.205	0.00
$\log(chng)$	58.5	<0.0005	1.18	0.084	0.00
$\log(FanOut)$	-5.8	<0.0005	0.34	0.067	0.00
Covrg	-3.5	0.001	-1.51	0.252	0.00

Increasing coverage demands exponentially increasing effort

Predicted levels of coverage for different numbers of changes to the test class and median Fan-Out of 7.



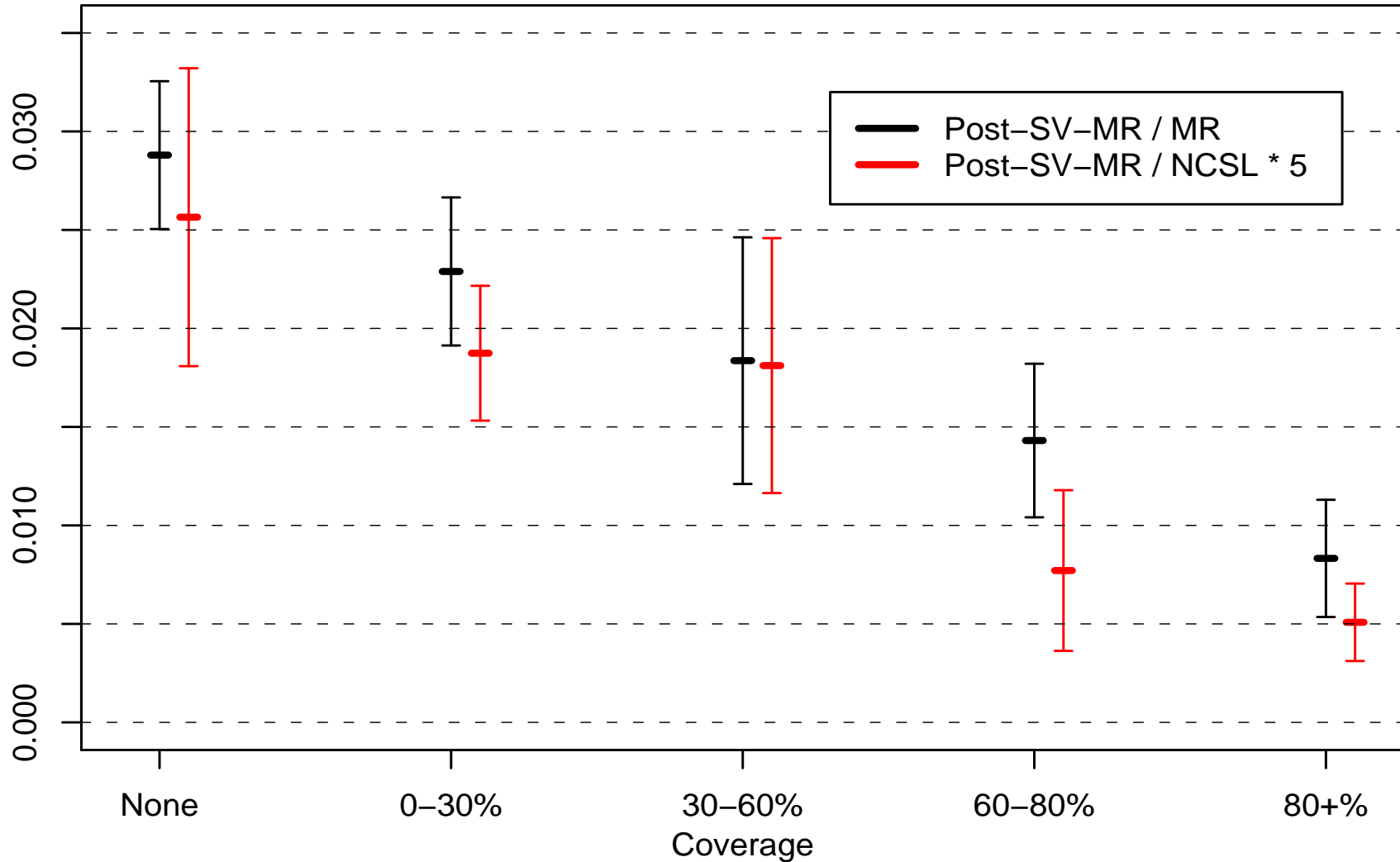
Validity

- ❖ Largest correlations among predictors: A: 0.25, M: 0.66
- ❖ Latent variables moderating coverage-defects relationship (Avaya)
 - ❖ Code functionality: e.g., UI has less coverage and more defects
 - ❖ Code ownership transition: i.e., offshoring lead to less coverage and more defects
 - ❖ Code complexity: harder to cover code gets less coverage
 - ❖ Developer experience: was associated with increased coverage and fewer defects

Summary

- ❖ Increase in coverage is associated with decrease in post-system-verification defects for all levels of coverage
- ❖ Increase in coverage is associated with increase in number of changes to associated test classes (exponentially increasing with the level of coverage)
- ❖ Coverage/defect relationship is moderated by functionality, code ownership transitions, code complexity, and developer experience

Increasing coverage decreases defect rate



Abstract

Test coverage is a promising measure of test effectiveness and development organizations are interested in cost-effective levels of coverage that provide sufficient fault removal with contained testing effort. We have conducted a multiple-case study on two dissimilar industrial software projects to investigate if test coverage reflects test effectiveness and to find the relationship between test effort and the level of test coverage. We find that in both projects the increase in test coverage is associated with decrease in field reported problems when adjusted for the number of pre-release changes. A qualitative investigation revealed several potential explanations, including code complexity, developer experience, the type of functionality, and remote development teams. All these factors were related to the level of coverage and quality, with coverage having an effect even after these adjustments. We also find that the test effort increases exponentially with test coverage, but the reduction in field problems increases linearly with test coverage. This suggests that for most projects the optimal levels of coverage are likely to be well short of 100%.

Bio

Audris Mockus

Avaya Labs Research

233 Mt. Airy Road

Basking Ridge, NJ 07920

ph: +1 908 696 5608, fax:+1 908 696 5402

<http://mockus.org>, <mailto:audris@mockus.org>,

picture:<http://mockus.org/images/small.gif>



Audris Mockus conducts research of complex dynamic systems. He designs data mining methods to summarize and augment the system evolution data, interactive visualization techniques to inspect, present, and control the systems, and statistical models and optimization techniques to understand the systems. Audris Mockus received B.S. and M.S. in Applied Mathematics from Moscow Institute of Physics and Technology in 1988. In 1991 he received M.S. and in 1994 he received Ph.D. in Statistics from Carnegie Mellon University. He works at Avaya Labs Research. Previously he worked at Software Production Research Department of Bell Labs.