





Towards Refined Code Coverage: A New Predictive Problem in Software Testing







Carolin Brandt, Aurora Ramírez

c.e.brandt@tudelft.nl aramirez@uco.es

		edium: >= 75 % high: >= 90 %	Functions: 1140	1294 <mark>8</mark>	76.6 % 38.1 %			
gradmac gradities 88.9 % (474/576) 40/45 (300.0 % (37.%) 67.5 % (300.0 % (37.%) 77.6 (300.0 % (37.%) Diving deeper on coverage gaps How to filter so that we focus on the relevant coverage gaps B) early return Exclude: A) 1-line coverage gaps B) early return ** **	base64.c basename.c cat.c chcon.c chorp.c chom.core.c chom.c	82.6 % 100.0 % 71.8 % 81.4 % 77.6 % 77.6 % 76.0 % 32.3 % 74.8 % 94.8 %	90/109 100.0 % 63/63 100.0 % 1068/234 100.0 % 107/7 100.0 % 152/196 100.0 % 152/196 100.0 % 169/202 55.6 % 76/100 100.0 % 51/155 50.0 % 49/66 100.0 %	5 / 5 4 / 4 6 / 6 2 / 6 3 / 3 6 / 6 5 / 9 2 / 2 3 / 6 3 / 3 5 / 5				
Codecov Percent of Repositories with Percent of Repositories with Percent of Repositories with	cp-hash.ccp.ccsplit.c	80.4 % 3 82.0 % 4	341 / 424 100.0 % 474 / 578 92.9 %	8/8 39/42 9/9	How to filte	er		
Percent of Percent of Repositories with Repositories with				E	Exclude:	A) 1-line c	overage gaps	, .
	Percent of Repositories v	with	Repositorie		1 1			"return"

Code Coverage

- 80% realistic to aim for
 - Mozilla Devs: "not worth the effort to add test"

Codecov:

- Test Quality over Test Quantity
- Engineering Time is Finite
- Not All Code is Equal

Pictures:

"Error"

https://interrupt.memfault.com/blog/testing-vs-overhead https://about.codecov.io/blog/the-case-against-100-code-coverage/

But which 80% should we cover?



Let's learn from open source projects!

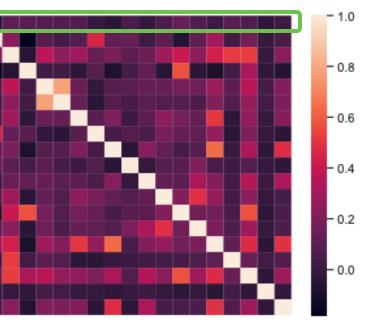
- Characterizing covered vs. not covered lines in Java projects
- Data Collection:
 - per-Line coverage (JaCoCo)
 - OO-metrics ("ck" by Mauricio Aniche)
 - AST syntax elements (/w tree-sitter)
- For this short paper: 1 Java project (allegro/hermes)



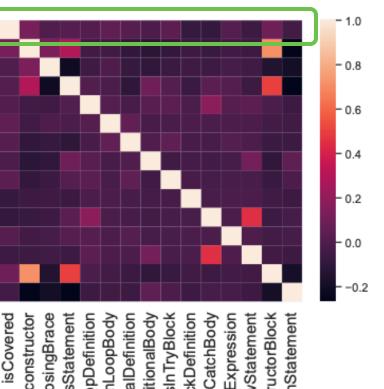
Can we apply machine learning?

• Do our metrics correlate with the target value (isCovered)?





isCovered returnsQty variablesQty parametersQty methodsInvokedLocalQty met.Inv.IndirectLocalQty loopQty comparisonsQty tryCatchQty parenthesizedExpsQty stringLiteralsQty numbersQty assignmentsQty mathOperationsQty maxNestedBlocksQty lambdasQty uniqueWordsQty modifiers logStatementsQty

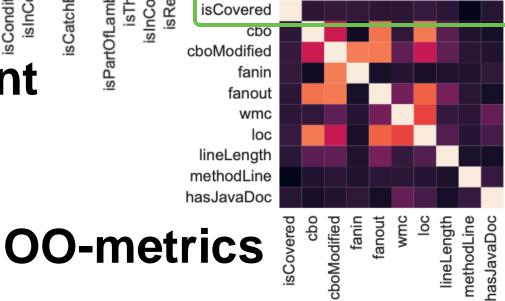


isCovered constructor isClosingBrace isStatement isLoopDefinition isInLoopBody isConditionalDefinition isInConditionalBody isInTryBlock isCatchBlockDefinition isInCatchBody isPartOfLambdaExpression isThrowStatement isInConstructorBlock isReturnStatement

> modifiers log Statements Qty uniqueWord Œ labl compariso ambd assignme paramet parenthesizedE stringLite oerat maxNestedBI met.Inv.Indirect mathO

Counting metrics

methodsInvoked -0.8 -0.6 -0.4 0.2 -0.0



isClosingBrace isStatement oopDefinition isConditionalDefinition sInConditionalBody isInTryBlock sCatchBlockDefinition is PartOfLambdaExpression SThrowStatement isInConstructorBlock isReturnStatement isInLoopBody sInCatchBody പ്പ Syntax element

Coverage vs.

TUDelft UNIVERSIDAD Ð CÓRDOBA

cboModified lineLength sCovered

6

Can we apply machine learning?

- No single metric correlates strongly (good!)
- Traditional ML can predict it well by combining metrics

Dataset is balanced 10k not vs 18k covered

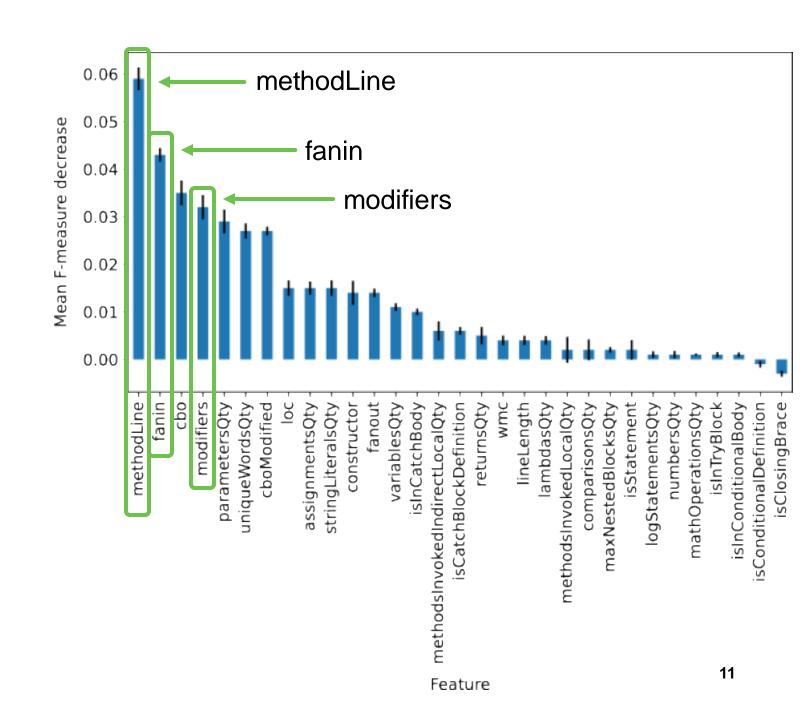
TABLE II: Classification results

Alg.	Accuracy	Bal. Accuracy	Precision	Recall	F-measure
DT	0.8930	0.8933	0.8701	0.8964	0.8830
kNN	0.7923	0.7897	0.7729	0.7633	0.7681
NB	0.5966	0.5779	0.5779	0.3888	0.4649
RF	0.9023	0.9018	0.8876	0.8968	0.8922



Important Features?

- Here: importance for random forest (strongest results)
- Interesting when sketching meaningful explanations!



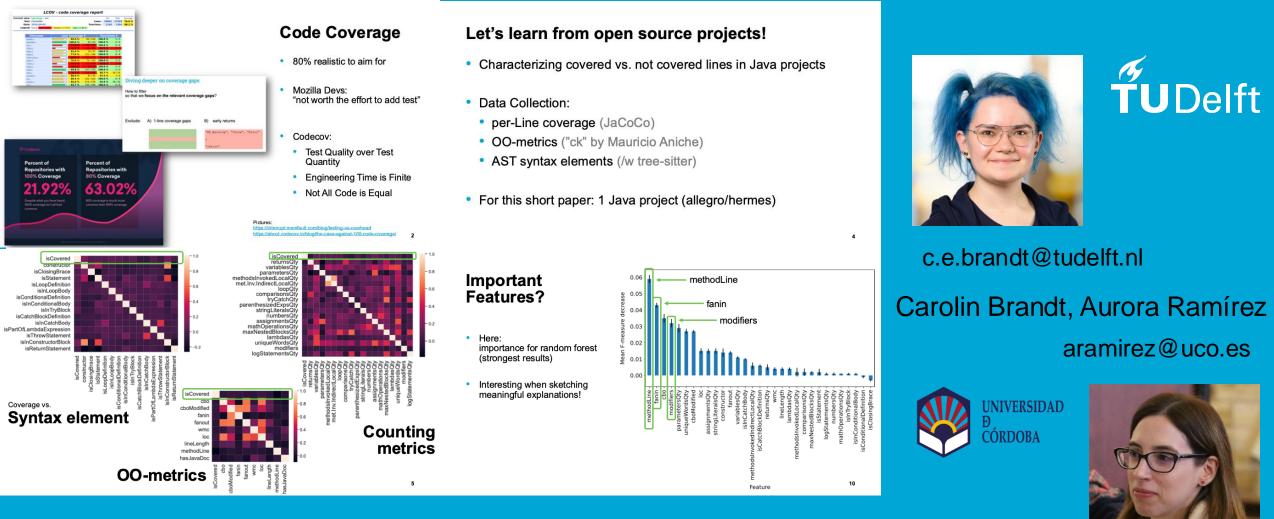


What is next?

- More projects (still Java / Open Source)
 - meaningful differences? cross-project learning feasible?

- Explainable AI: Can we create meaningful explanations to developers on why this code should be tested?
- Ultimately: Build developer-driven coverage metric







Towards Refined Code Coverage: A New Predictive Problem in Software Testing