

An Empirical Study of Bugs in Test Code

The 31st International Conference on
Software Maintenance and Evolution(ICSME2015)



Arash Vahabzadeh



Amin Milani Fard



Ali Mesbah

University of British Columbia Vancouver, BC, Canada

- テストにテストコードを書くことが一般的になってきている
 - リグレッションの確認にも有効

ソースコード

```

1 package travisTest;
2
3 public class Car {
4     private String no;
5     public Car(String no){
6         this.no = no;
7     }
8
9     public void run(){
10        System.out.println("ぶううううん");
11    }
12
13    public String getNo(){
14        return no;
15    }

```

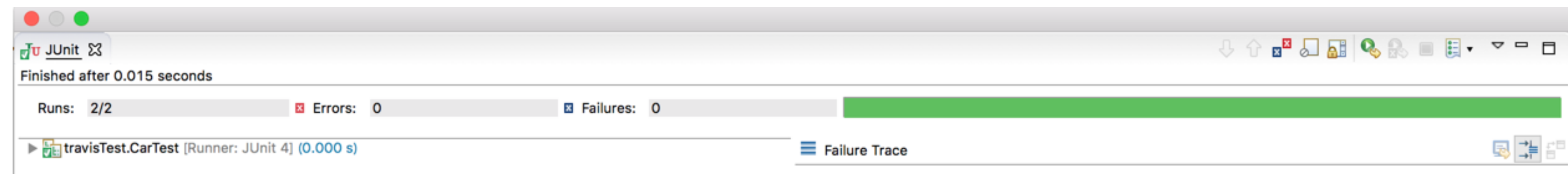
テストコード

```

1 package travisTest;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Before;
6 import org.junit.Test;
7
8 public class CarTest {
9     Car car;
10
11     @Before
12     public void init(){
13         car = new Car("わ 123-456");
14     }
15
16     @Test
17     public void testGetNo_S001() {
18         assertEquals(car.getNo(),"わ 123-456");
19     }

```

テスト結果



背景

- ソフトウェアの大規模化によりテストコードも大規模化
 - テストコードは人が作成するもの⇒**バグを含んでいる可能性**
 - テストコードのバグについて実証的分析はされていない
 - 出現頻度, インパクト, 原因
 - テストコードのバグは大きく 2 つに分けられると考えられる[1]
 - » プロダクトコードに誤りが**ある**が, テストは**成功** (Silent horrors)
 - » プロダクトコードに誤りが**ない**が, テストは**失敗** (False alarms)

Research Questions

RQ1. テストコードのバグは、 どの程度出現するか？

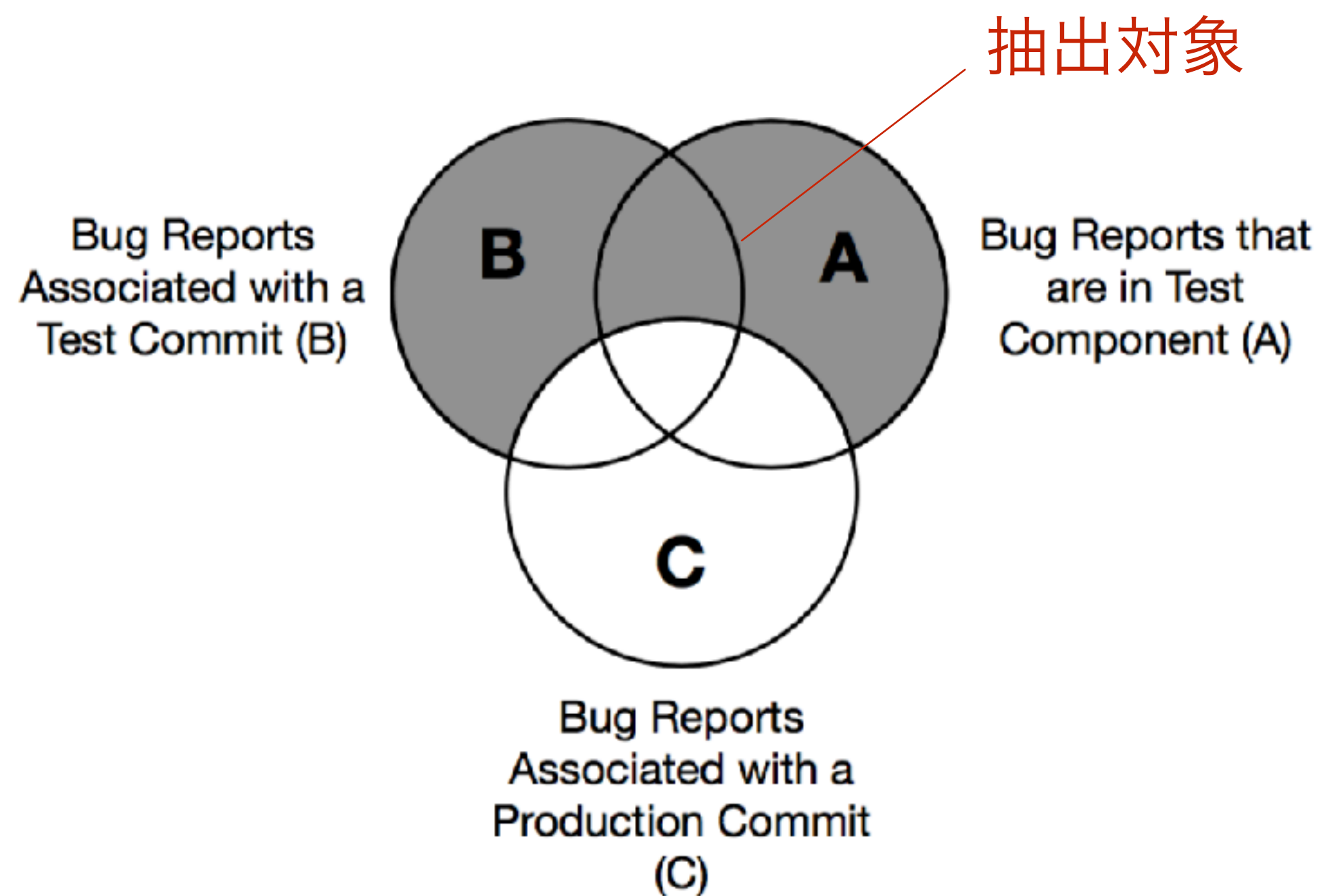
RQ2. テストコードのバグは、 どのような種類が一般的か？

RQ3. テストバグは、 製品のバグとは別に扱われるか？

RQ4. テストバグの発見に、 バグ検出ツールは利用できるか？

テストバグの抽出

- 2タイプのテストバグを抽出

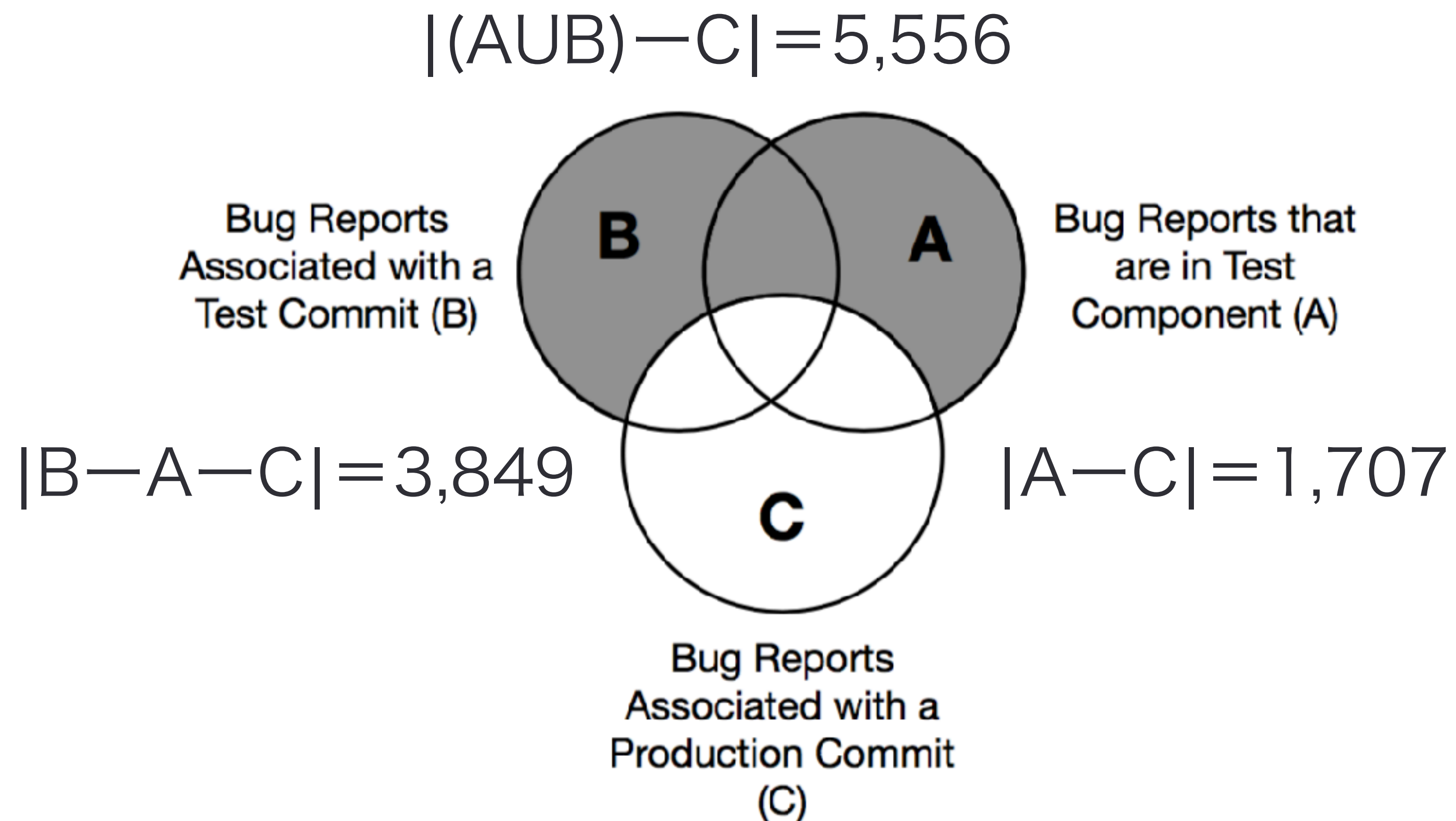


- ASFの110プロジェクトからデータを利用
 - サブプロジェクト数：448

RQ1：どの程度出現するか？

6

- 47%のサブプロジェクトでテストバグが発見 (211/448)



合計	5,556
平均値	12.4
中央値	0
最大値	614
最小値	0
標準偏差	48.3

RQ1：どの程度出現するか？（詳細）

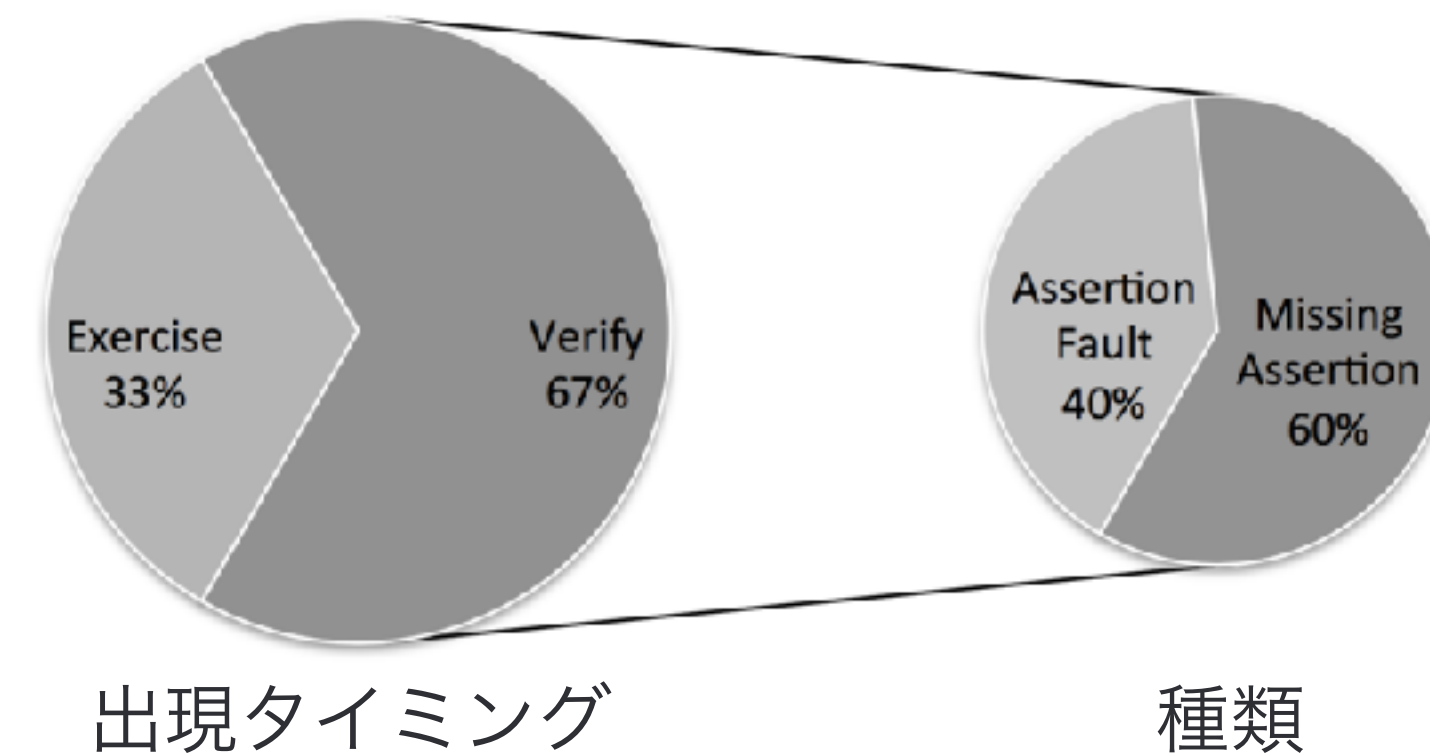
- 47%のサブプロジェクトでテストバグが発見 (211/448)

テストバグが多いプロジェクトTOP10

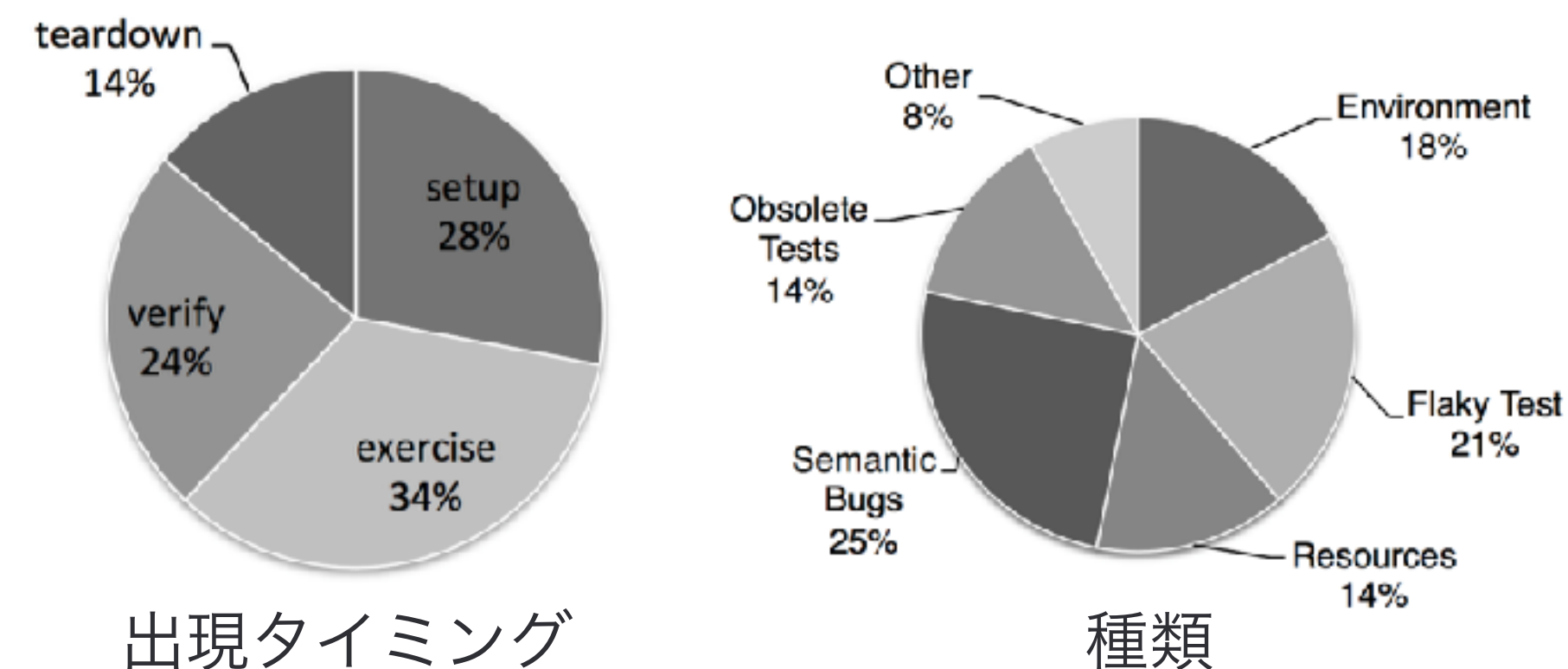
Project	Production Code KLOC	Test Code KLOC	# Test Bug Reports
Derby	386	370	614
HBase	587	195	440
Hive	836	124	295
Hadoop HDFS	101	57	286
Hadoop Common	1249	380	279
Hadoop Map/Reduce	60	24	231
Accumulo	405	78	187
Qpid	553	93	152
Jackrabbit Content Repository	247	107	145
CloudStack	1361	228	111

RQ 2 : どんな種類が一般的か？

- 499件 (9%) の目視を実施
 - 3%の不具合が, **Silent horrors** (プロダクトコードに誤りがあるが, テストは**成功**)



- 97%の不具合が, **False alarms** (プロダクトコードに誤りが**ない**が, テストは**失敗**)



RQ3 : テストバグは特別に扱われるか？

9

- 製品バグとテストバグで，優先度や修正時間などを比較

Metric	Type	Med	Mean	SD	Max	d	OR	p-value
Priority	PR	3.00	2.91	0.76	5.00	-0.13	0.78	4.9e-14
	TE	3.00	2.80	0.75	5.00			
Resolution Time(days)	PR	6.39	109.70	282.04	2843.56	-0.20	0.69	<2.2e-16
	TE	2.77	58.97	213.72	2666.60			
#Comments	PR	3.00	4.91	6.74	101.00	0.15	1.31	
	TE	4.00	5.88	6.26	99.00			
#Authors	PR	2.00	2.41	1.53	18.00	0.31	1.77	
	TE	2.00	2.89	1.53	12.00			
#Watchers	PR	0.00	1.32	2.04	24.00	0.25	1.58	
	TE	1.00	1.84	2.06	16.00			

- 開発者は，テストバグの方が活発に貢献している
- テストバグの方が修正時間が短い

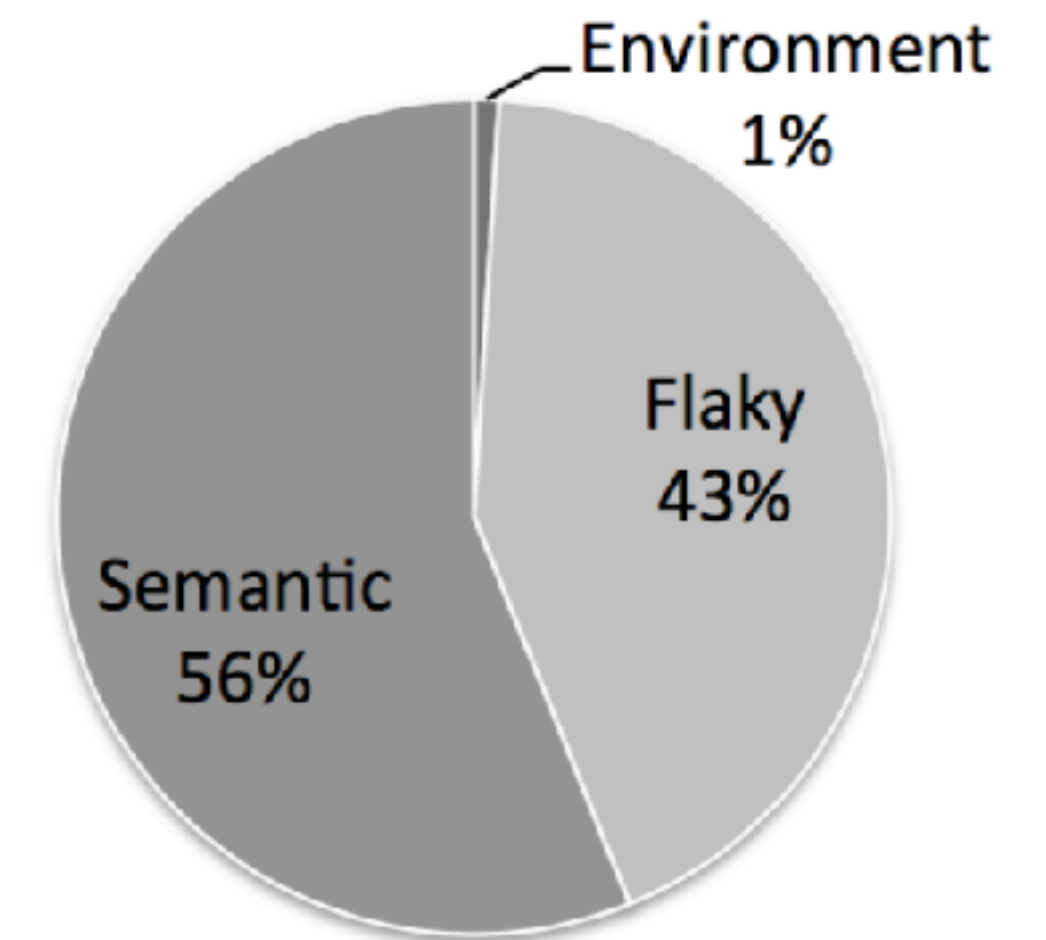
RQ4：バグ検出ツールは利用できるか？

10

- 20プロジェクトのテストコードにFindBugsを適用

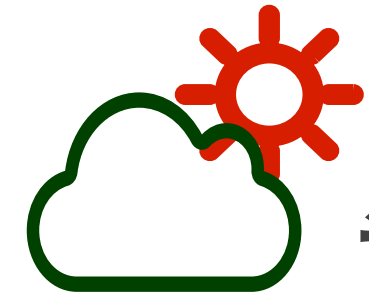
Min	Mean	Median	σ	Max	Total
0	1.6	0	5.5	48	205

Bug Description	Bug Category	Percentage
Inconsistent synchronization	Flaky	29.8%
Possible null pointer dereference in method on exception path	Semantic	17.6%
Using pointer equality to compare different types	Semantic	8.8%
Possible null pointer dereference	Semantic	7.3%
Class defines field that masks a superclass field	Semantic	3.9%
Nullcheck of value previously dereferenced	Semantic	2.9%
An increment to a volatile field isn't atomic	Flaky	2.9%
Method call passes null for nonnull parameter	Semantic	2.4%
Incorrect lazy initialization and update of static field	Flaky	2.4%
Null value is guaranteed to be dereferenced	Semantic	2.0%



- 不具合50件(623件)を目視で確認したところ、Findbugsの警告箇所がバグになっていない

所感

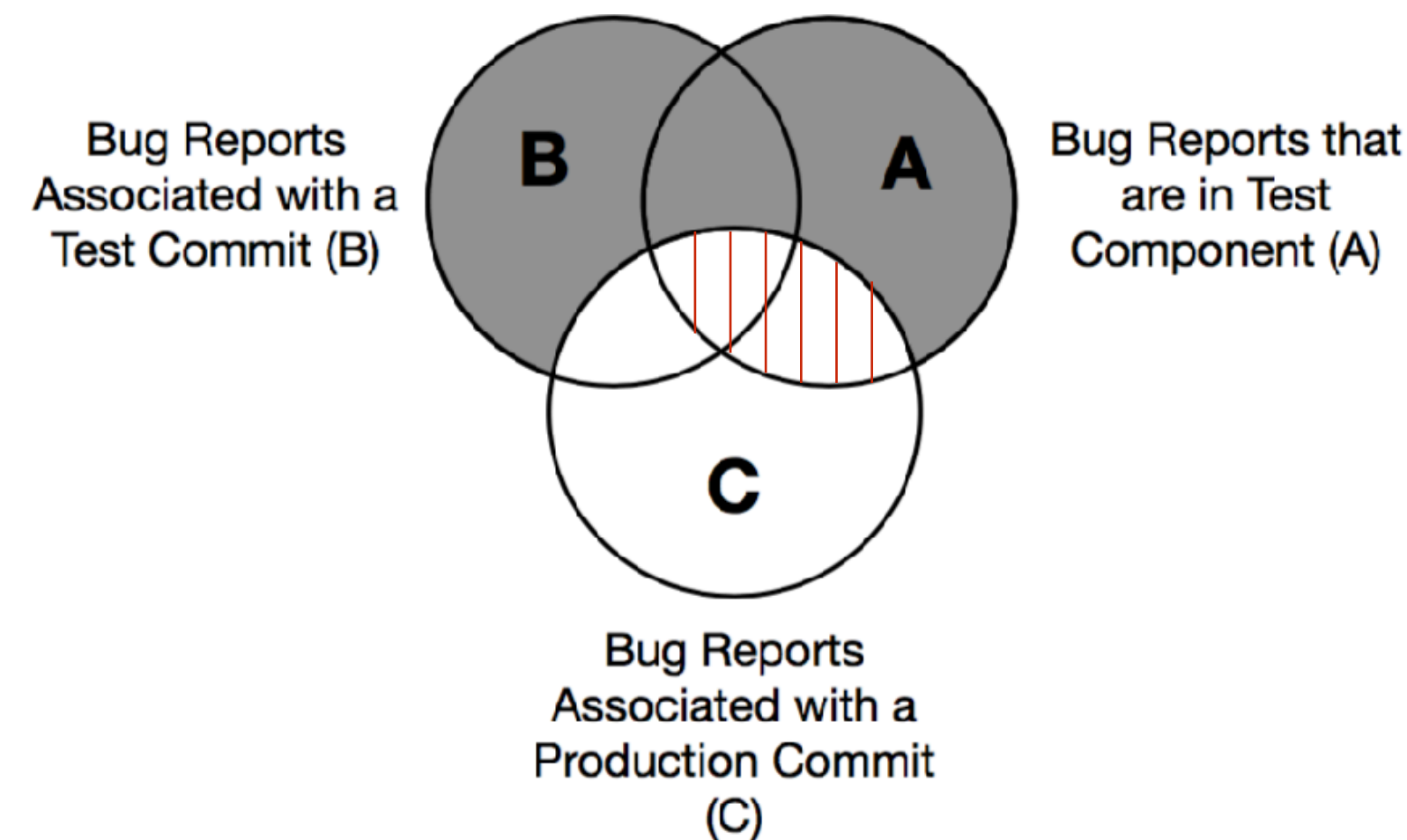


先にやられてた. . . . でもやっぱり重要な研究



一番大事なところをやってない！？

プロダクトコードの誤りのために発生した不具合



目視したけど、テストバグが1プロジェクトに
数百件も無い気がする. . . . (そもそも抽出方法が微妙)