ICSE2023

Measuring and Mitigating Gaps in Structural Testing

Soneya Binta Hossain

Matthew Dwyer

Sebastian Elbaum

Anh Nguyen-Tuong

LESS LAB
LEADING ENGINEERING FOR SAFE SOFTWARE

University of Virginia

Measuring Gaps

	E	E∩C	G
1:	✓		✓
2:	✓		✓
3:	✓		✓
4:	✓	✓	
5:	✓		✓
6:	✓		✓
7:	✓		✓
8:	✓	✓	

```
public class Triangle {

      int s1, s2, s3, p, color;
      Triangle(int a1, int a2, int a3, int c) {
1:         s1 = a1;
2:         s2 = a2;
3:         s3 = a2;🐞
4:         color = c;
5:         setPerimeter();
      }

      private void setPerimeter() {
6:         p = s1 + s2 + s3;
      }

      public int getPerimeter() {
7:         return p;
      }

      public int getColor() {
8:         return color;
      }
}
```

```
@Test
public void testColor() {
  Triangle t = new Triangle(2,3,2,1);
  t.getPerimeter();
  assertEquals(1, t.getColor());
}
```

Covered: 100%

Checked: 25%

In Gap: 75%

Mitigating Gaps

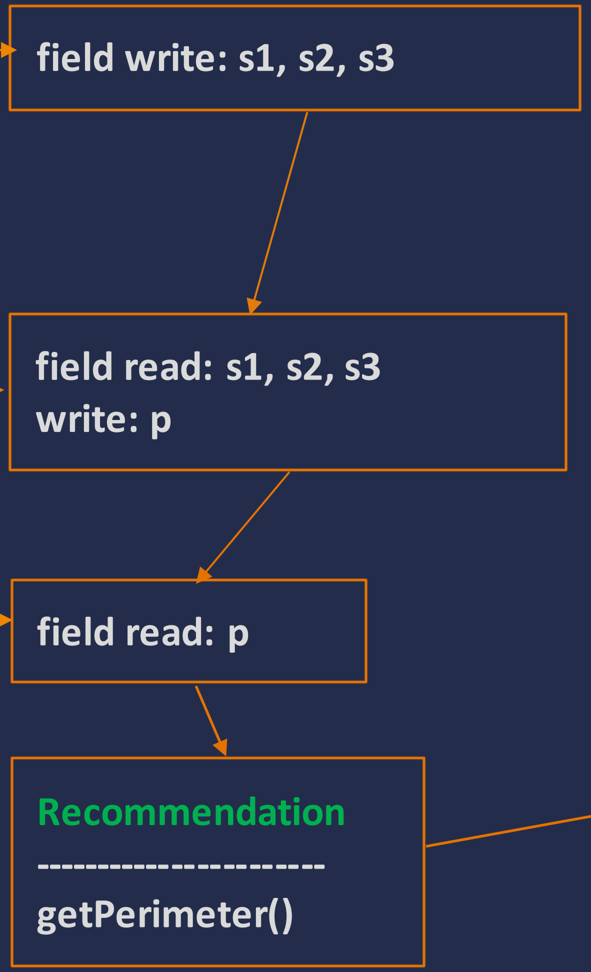
```
public class Triangle {

      int s1, s2, s3, p, color;
      Triangle(int a1, int a2, int a3, int c) {
1:          s1 = a1;
2:          s2 = a2;
3:          s3 = a2;
4:          color = c;
5:          setPerimeter();
      }

      private void setPerimeter() {
6:           p = s1 + s2 + s3;
      }

      public int getPerimeter() {
7:           return p;
      }

      public int getColor() {
8:           return color;
      }
}
```



field write: s1, s2, s3

field read: s1, s2, s3
write: p

field read: p

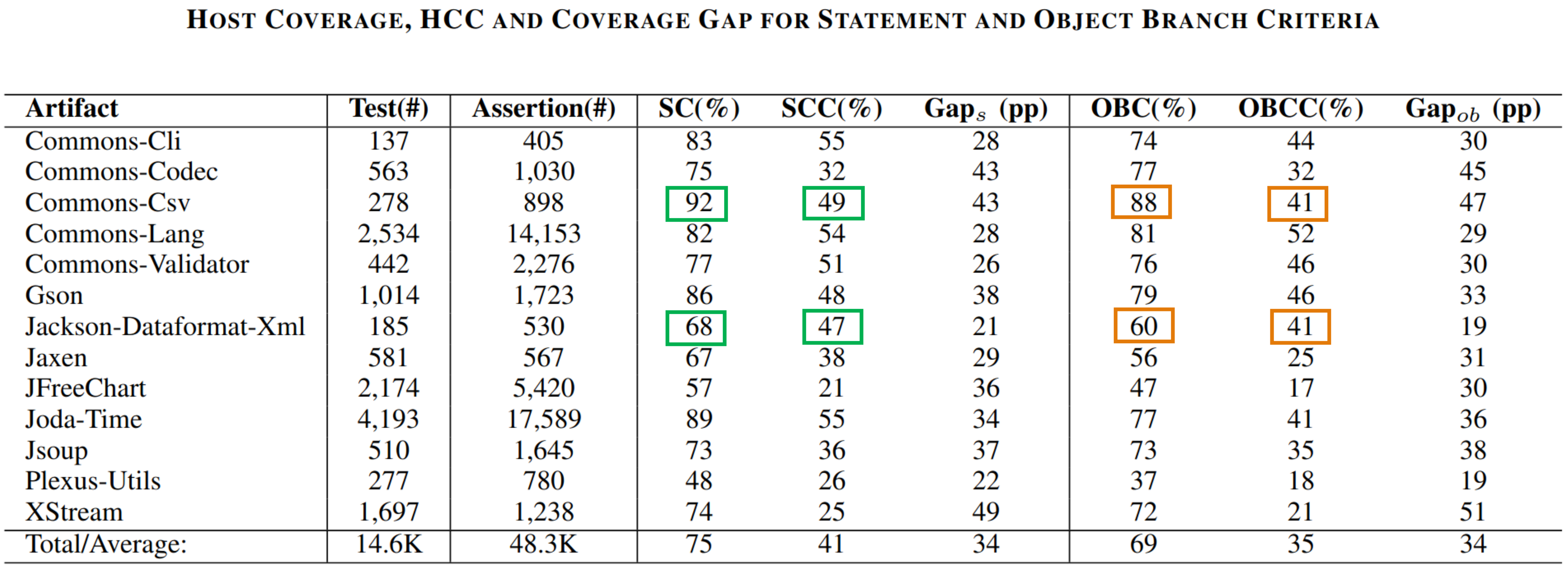
Recommendation
----
getPerimeter()

```
@Test
public void testColor() {
  Triangle t = new Triangle(2,3,2,1);
  assertEquals(1,  t.getColor());
  assertEquals(7,t.getPerimeter());
}
```

Evaluation:

Artifacts

- 13 Java Applications
- 16K tests
- 51.6K test assertions



HOST COVERAGE, HCC AND COVERAGE GAP FOR STATEMENT AND OBJECT BRANCH CRITERIA

Artifact	Test(#)	Assertion(#)	SC(%)	SCC(%)	Gap_s (pp)	OBC(%)	OBCC(%)	Gap_{ob} (pp)
Commons-Cli	137	405	83	55	28	74	44	30
Commons-Codec	563	1,030	75	32	43	77	32	45
Commons-Csv	278	898	92	49	43	88	41	47
Commons-Lang	2,534	14,153	82	54	28	81	52	29
Commons-Validator	442	2,276	77	51	26	76	46	30
Gson	1,014	1,723	86	48	38	79	46	33
Jackson-Dataformat-Xml	185	530	68	47	21	60	41	19
Jaxen	581	567	67	38	29	56	25	31
JFreeChart	2,174	5,420	57	21	36	47	17	30
Joda-Time	4,193	17,589	89	55	34	77	41	36
Jsoup	510	1,645	73	36	37	73	35	38
Plexus-Utils	277	780	48	26	22	37	18	19
XStream	1,697	1,238	74	25	49	72	21	51
Total/Average:	14.6K	48.3K	75	41	34	69	35	34

Finding: Gaps range from 19-51 percentage points (pp), with an average of 35pp

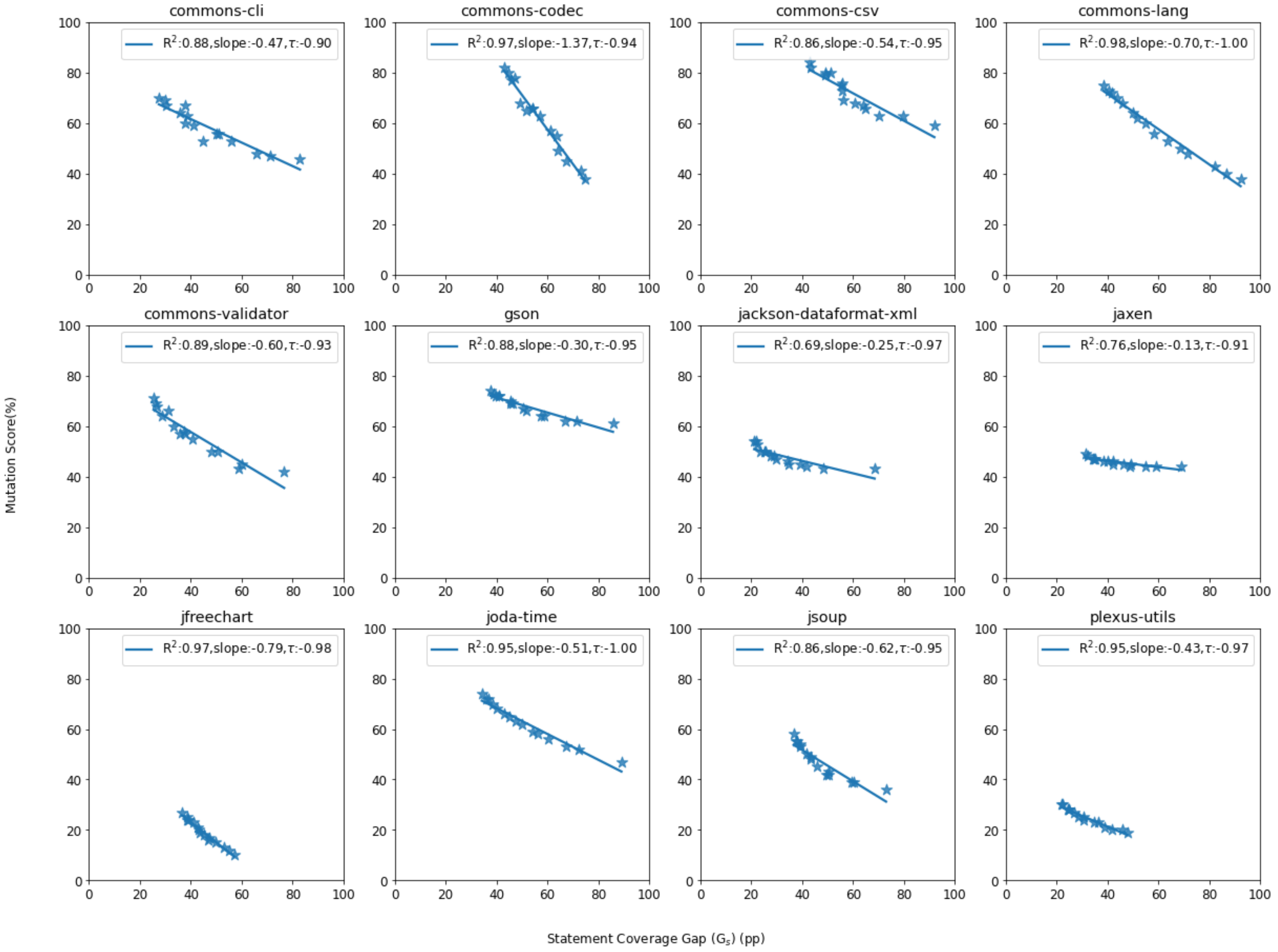
Impact of Gaps on Fault Detection

Granularity: Application, Package

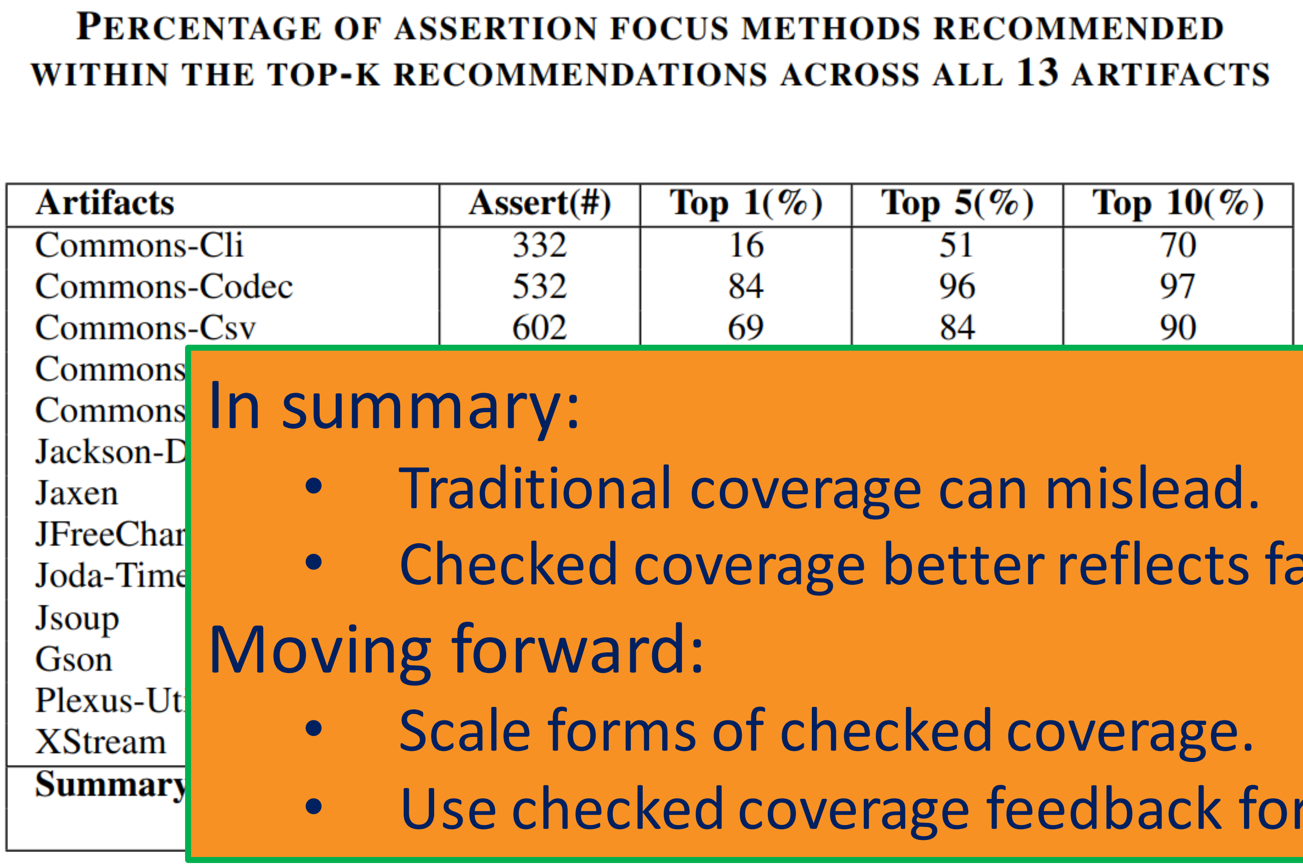
Criteria: Statement, Object branch

Findings:

- Gap size and fault-detection effectiveness have a statistically significant, strong negative correlation



Recommender Performance



PERCENTAGE OF ASSERTION FOCUS METHODS RECOMMENDED WITHIN THE TOP-K RECOMMENDATIONS ACROSS ALL 13 ARTIFACTS

Artifacts	Assert(#)	Top 1(%)	Top 5(%)	Top 10(%)
Commons-Cli	332	16	51	70
Commons-Codec	532	84	96	97
Commons-Csv	602	69	84	90
Commons				
Commons				
Jackson-D				
Jaxen				
JFreeChar				
Joda-Time				
Jsoup				
Gson				
Plexus-Ut				
XStream				
Summary				

In summary:

- Traditional coverage can mislead.
- Checked coverage better reflects fault detection.

Moving forward:

- Scale forms of checked coverage.
- Use checked coverage feedback for test suite improvement.

We applied the recommender to Joda-time, resulting in up to 57pp and an average of 13pp improvement in fault detection effectiveness.