The Impact of Equivalent Mutants

Bernhard J. M. Grün
Saarland University

with David Schuler and Andreas Zeller

Equivalent Mutants

A mutation may not change the semantics of a program. Then it is equivalent.

- What is the problem with them?
- Is it hard to find them?
- How frequent are these?
- How can we get rid of them?

Is this mutation equivalent?

```
if(!map.containsKey(key)) {
    Integer value = key.length();
    map.put(key, value);
}
```

Is this mutation equivalent?

```
                                    true
if(!map.containsKey(key)) {
    Integer value = key.length();
    map.put(key, value);
}
```

Is this mutation equivalent?

```
                                    true
if(!map.containsKey(key)) {
    Integer value = key.length();
    map.put(key, value);
}
```

Equivalent

Is this mutation equivalent?

```
counter++;
if(!map.containsKey(key)) {
    Integer value = counter;
    map.put(key, value);
}
```

Is this mutation equivalent?

```
counter++;
if(!map.containsKey(key)) {
    Integer value = counter;
    map.put(key, value);
}
```

true

Is this mutation equivalent?

```
counter++;
if(!map.containsKey(key)) {
    Integer value = counter;
    map.put(key, value);
}
```

true

Not Equivalent

Background

- Baldwin and Seyward: Heuristic Approach
  - Semantic-preserving compiler optimization
  - Offutt and Craft: 10 % of equivalent mutants
- Offutt and Pan: Path Conditions
  - If constraint solver shows subsequent states are equivalent then mutant is equivalent
  - 48 % of equivalent mutants
  - 11 Fortran-77 Programs (10-30 Statements)

Javalanche

Javalanche

An efficient mutation framework for Java

Javalanche

An efficient mutation framework for Java

- Manipulates byte code directly

Javalanche

An efficient mutation framework for Java

- Manipulates byte code directly
- Implements *selective mutation:*
  - replace constant *C* by *C* ±1, or 0
  - negate branch condition
  - replace operators (+ by -, * by /, etc.)

Javalanche

An efficient mutation framework for Java

- Manipulates byte code directly
- Implements *selective mutation:*
  - replace constant *C* by *C* ±1, or 0
  - negate branch condition
  - replace operators (+ by -, * by /, etc.)
- Uses coverage and mutant schemata

Javalanche: Efficiency

Javalanche: Efficiency

Jaxen	12,449 LOC	6,626 mutations	6.5h

Javalanche: Efficiency

Jaxen	12,449 LOC	6,626 mutations	6.5h
XStream	14,480 LOC	5,186 mutations	6h

Javalanche: Efficiency

Jaxen	12,449 LOC	6,626 mutations	6.5h
XStream	14,480 LOC	5,186 mutations	6h
AspectJ	94,902 LOC	47,146 mutations	14h

Javalanche: Efficiency

Jaxen	12,449 LOC	6,626 mutations	6.5h

✓ We are able to analyze real world programs

AspectJ	94,902 LOC	47,146 mutations	14h

Javalanche: Efficiency

Jaxen	12,449 LOC	6,626 mutations	6.5h

✓ We are able to analyze real world programs

AspectJ	94,902 LOC	47,146 mutations	14h

Publicly available this Summer

Equivalent Mutants

Experimental base – the *Jaxen XQuery Engine:*

- 6626 mutations applied by Javalanche
- 29% undetected by test suite

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	
equivalent	
undecided	

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

40% equivalent mutants!

Manual Classification

- About 15 minutes per mutant to classify.
  - Sometimes up to 2 hours per mutant.
- For 1900 (29%) mutations in Jaxen:
  - Nearly 500 hours!

Manual Classification

Idea: Measuring Impact

Coverage Impact

Original

Mutant

Coverage Impact

A mutated loop

Original

Mutant

```
for (int i = 0; i < 10; i++)
```

Coverage Impact

A mutated loop

Original

```
for (int i = 0; i < 10; i++)
```

10
10
10
10
10

Mutant

Coverage Impact

A mutated loop

Original

```
for (int i = 0; i < 10; i++)
```

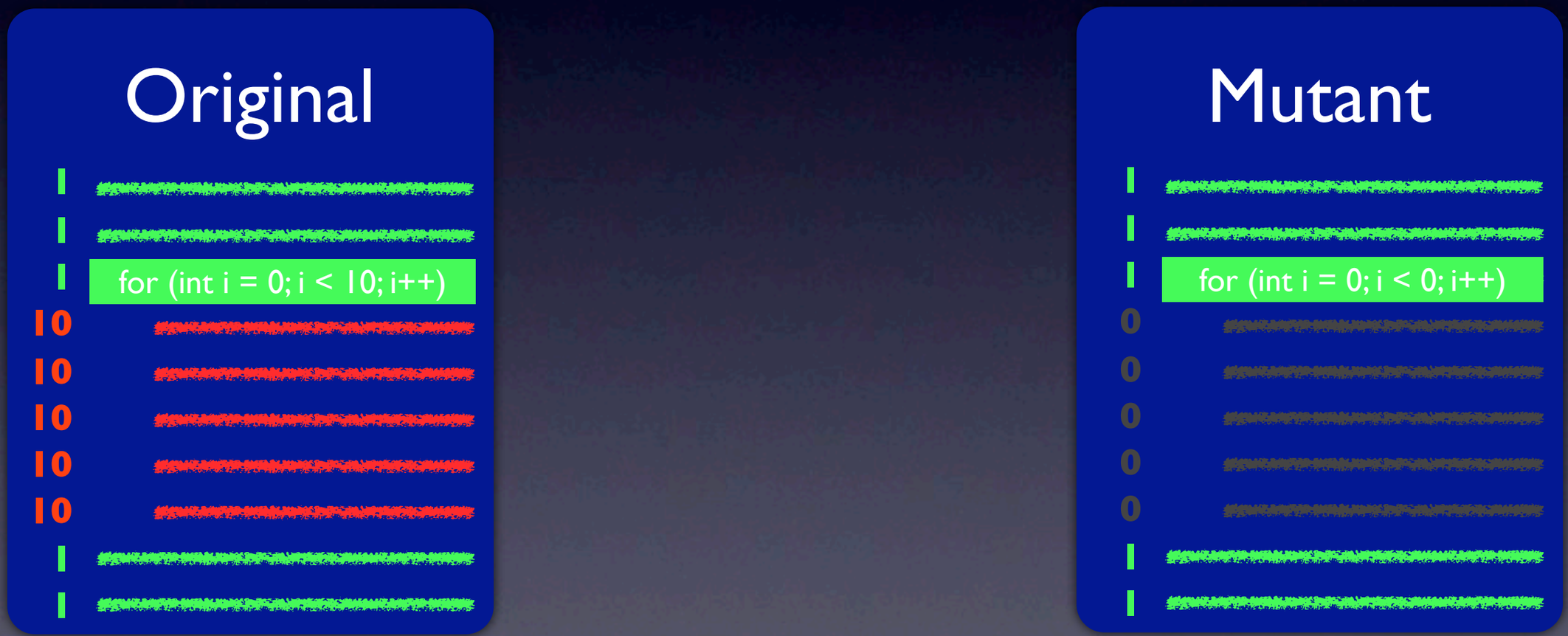
10 10 10 10 10

Mutant

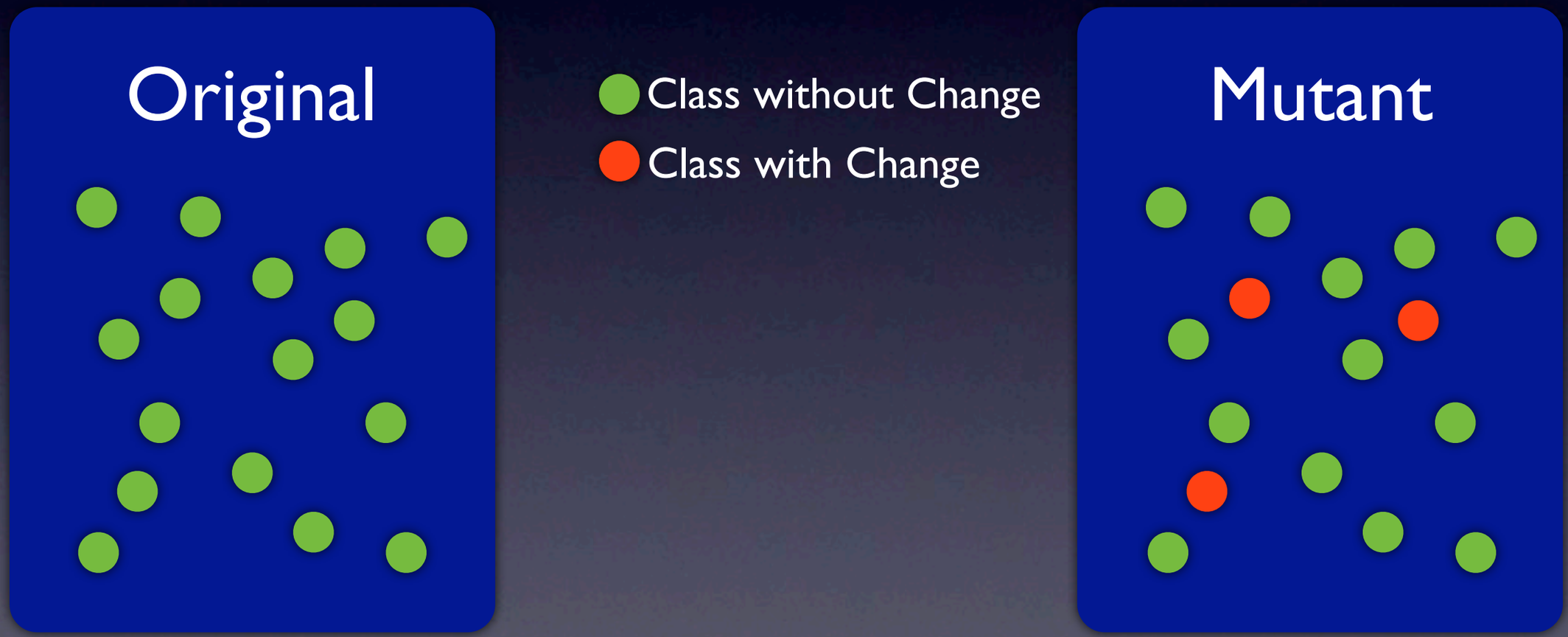
```
for (int i = 0; i < 0; i++)
```

Coverage Impact

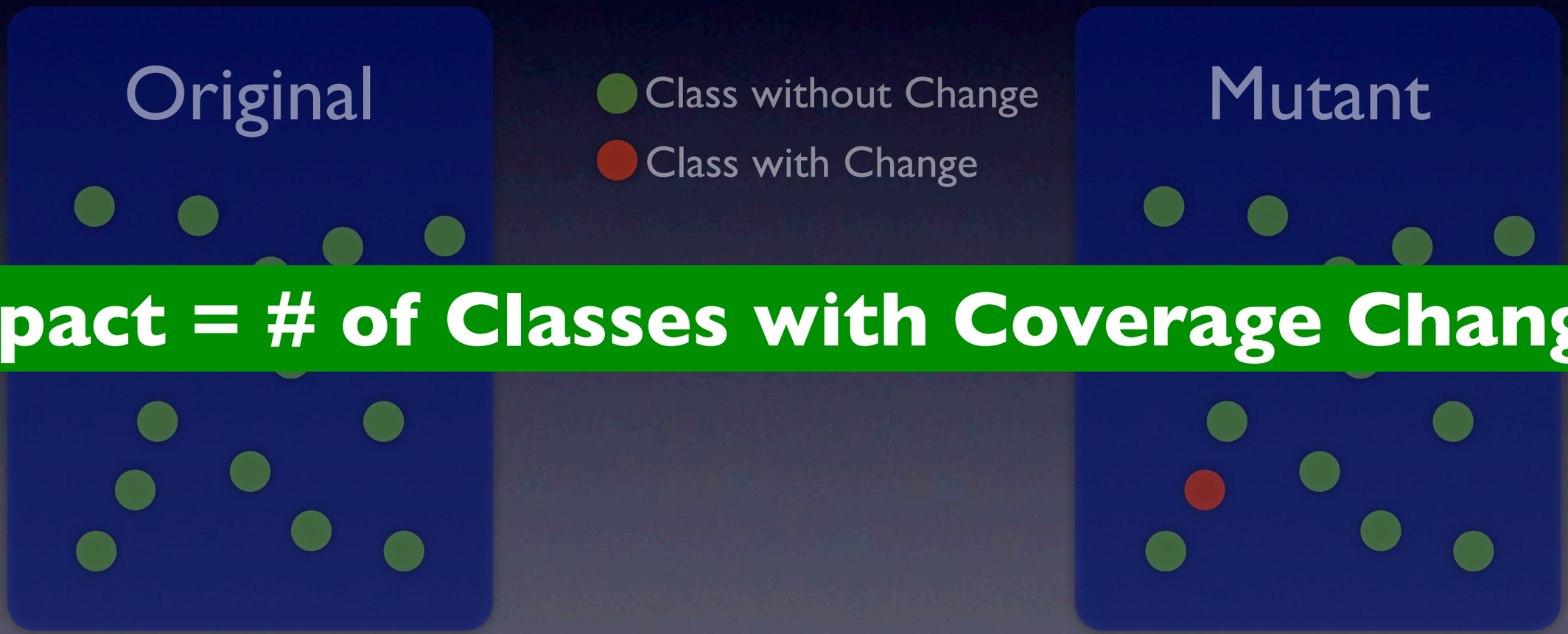
A mutated loop



Coverage Impact

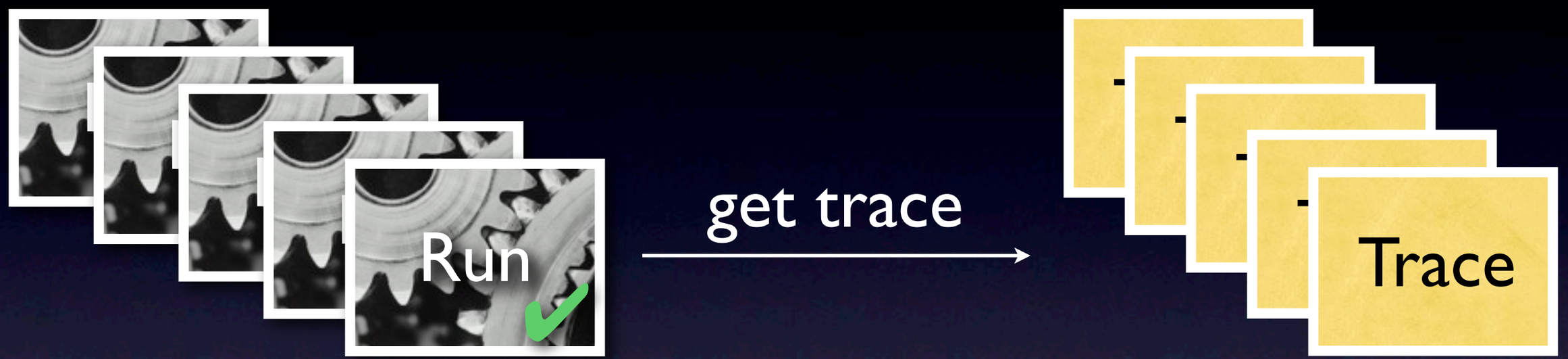


Coverage Impact

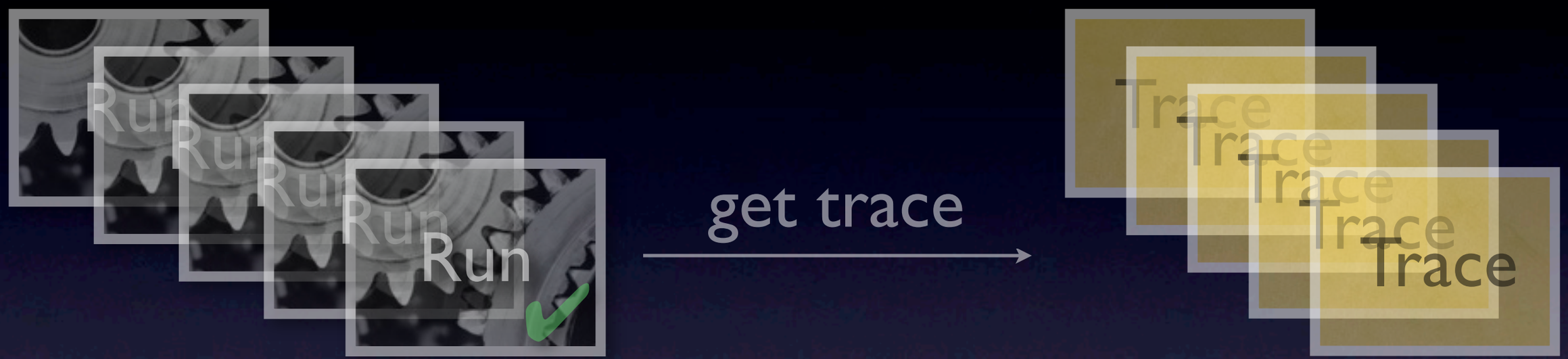


Impact = # of Classes with Coverage Changes

Tracing Programs



Tracing Programs



Original

compare traces

Mutant

Quality of Impact

- Using the same 20 random mutations

	impact	no impact
non-equivalent		
equivalent		
undecided		

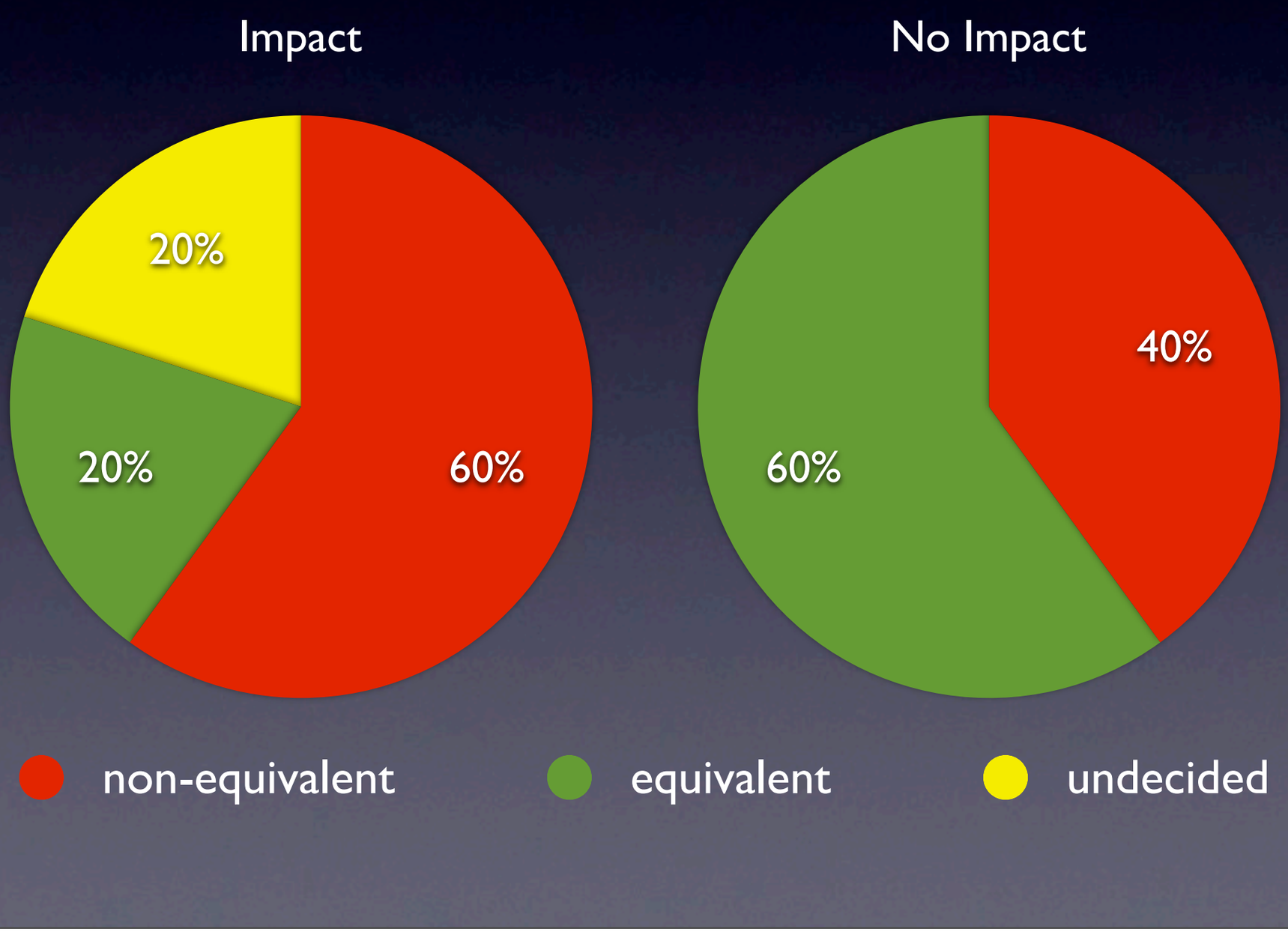
Quality of Impact

- Using the same 20 random mutations

	impact	**no impact**
non-equivalent	6	4
equivalent	2	6
undecided	2	0

Quality of Impact

- Using the same 20 random mutations



Ranking along Impact

- Hypothesis:
  - Higher Impact means more likely non-equivalent.
- Experiment:
  - 20 mutations with the *most impact*.
  - 20 mutations with the *least impact*.

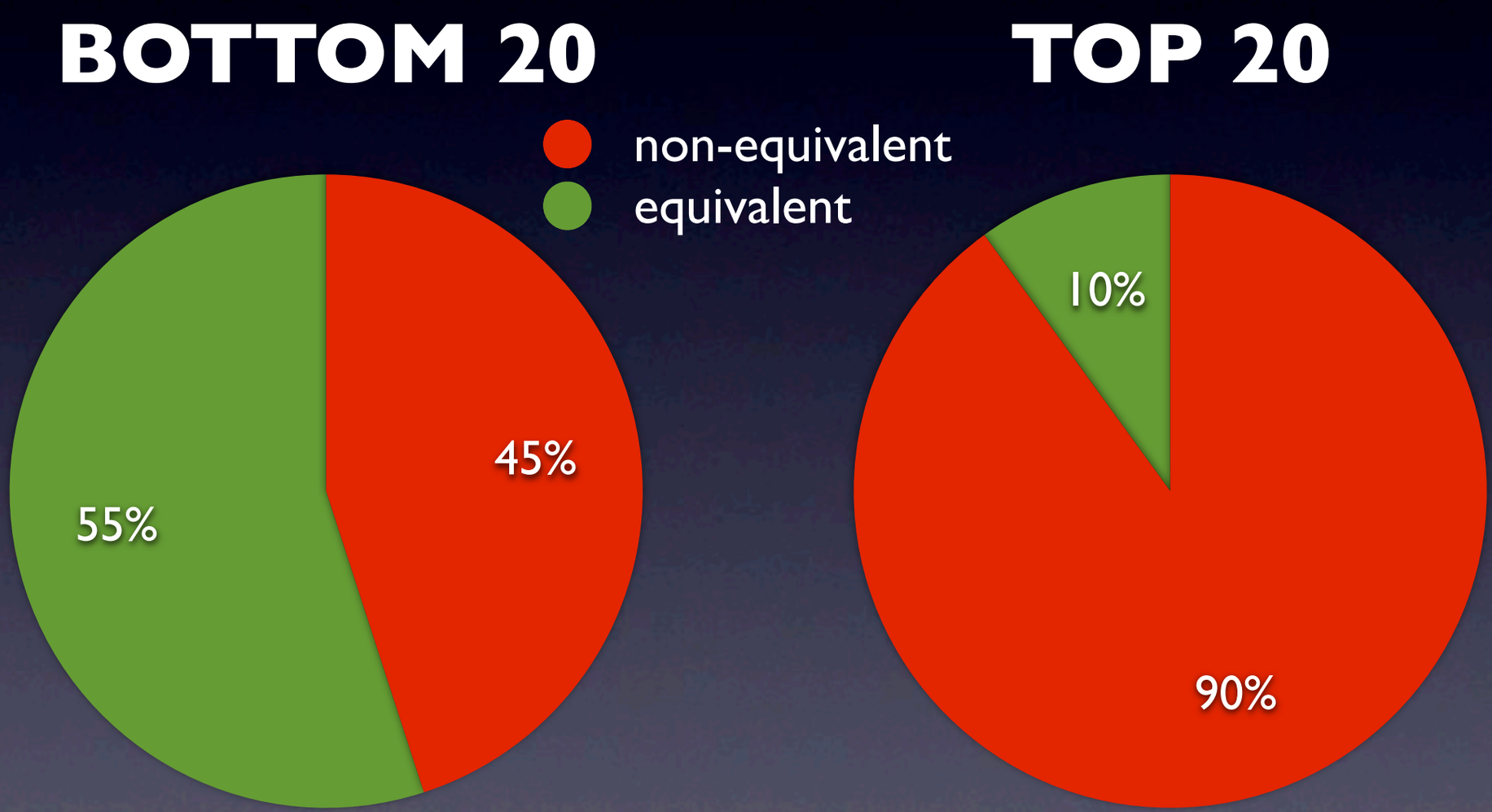
Ranking Example

IMPACT	MUTATION
10	A
2	B
0	C
3	D
7	E
0	F
5	G

Ranking Example

IMPACT	MUTATION
10	A
7	E
5	G
3	D
2	B
0	C
0	F

Ranking Results



Ranking along Impact

String Alteration

```
// org.jaxen.XPathSyntaxException, Line 140
public String getPositionMarker() {
    StringBuffer buf = new StringBuffer();
    int pos = getPosition();
    for (int i = 0; i < pos; i++) {
        buf.append(“ ”);
    }
    buf.append(“^”);
    return buf.toString();
}
```

String Alteration

```
// org.jaxen.XPathSyntaxException, Line 140
public String getPositionMarker() {
    StringBuffer buf = new StringBuffer();
    int pos = getPosition();
    for (int i = 0; i < pos; i++) {
        buf.append(“ ”);
    }
    buf.append(“^”);
    return buf.toString();
}
```

Return Values

```
// org.jaxen.function.CeilingFunction, Line 129
public static Double evaluate(Object obj, Navigator nav) {
    Double value = NumberFunction.evaluate(obj, nav);

    return new Double( Math.ceil( value.doubleValue() ) );
}
```

Return Values

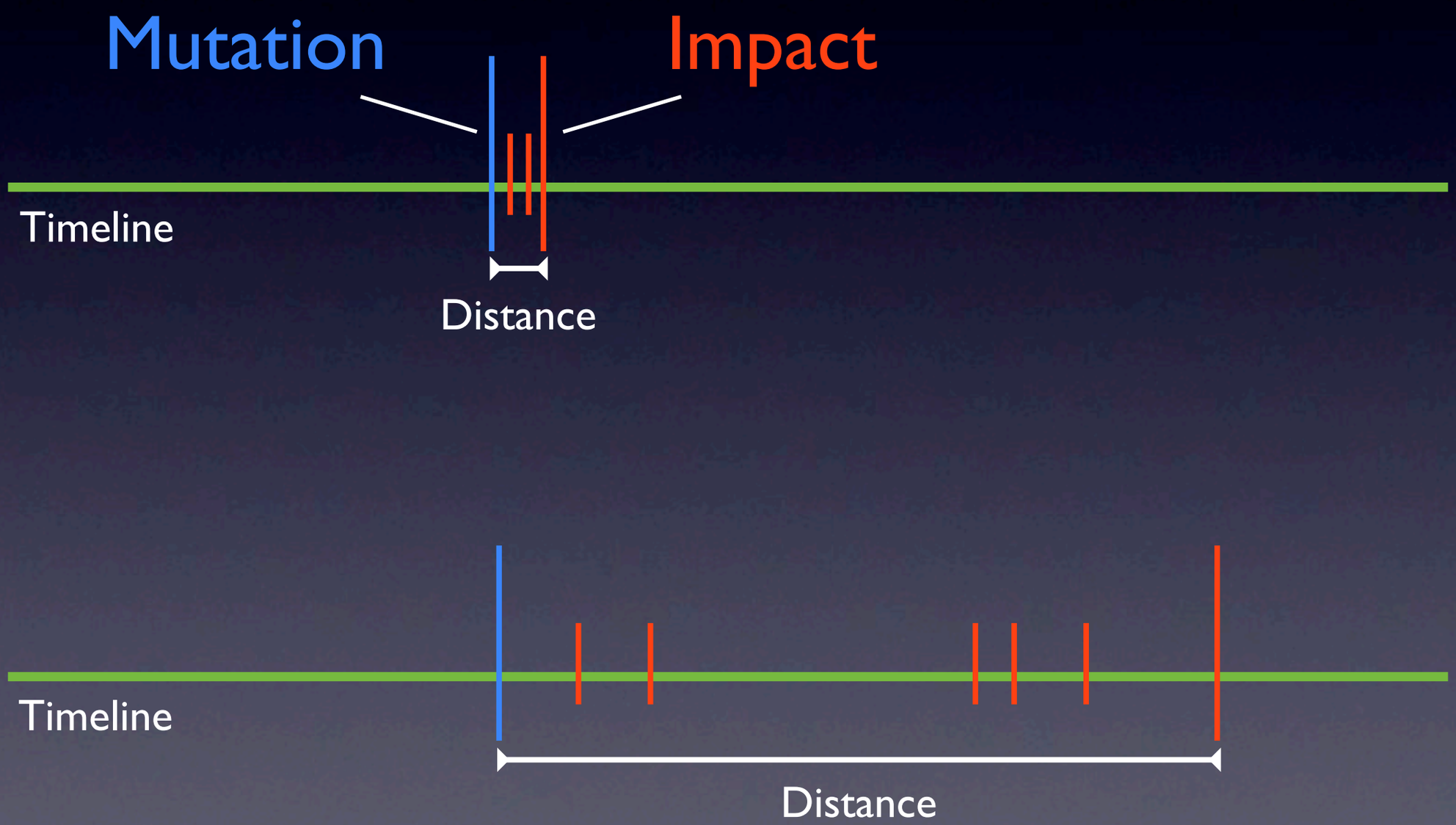
```
// org.jaxen.function.CeilingFunction, Line 129
public static Double evaluate(Object obj, Navigator nav) {
    Double value = NumberFunction.evaluate(obj, nav);

    return new Double( Math.ceil( value.doubleValue() ) );
}
```

0

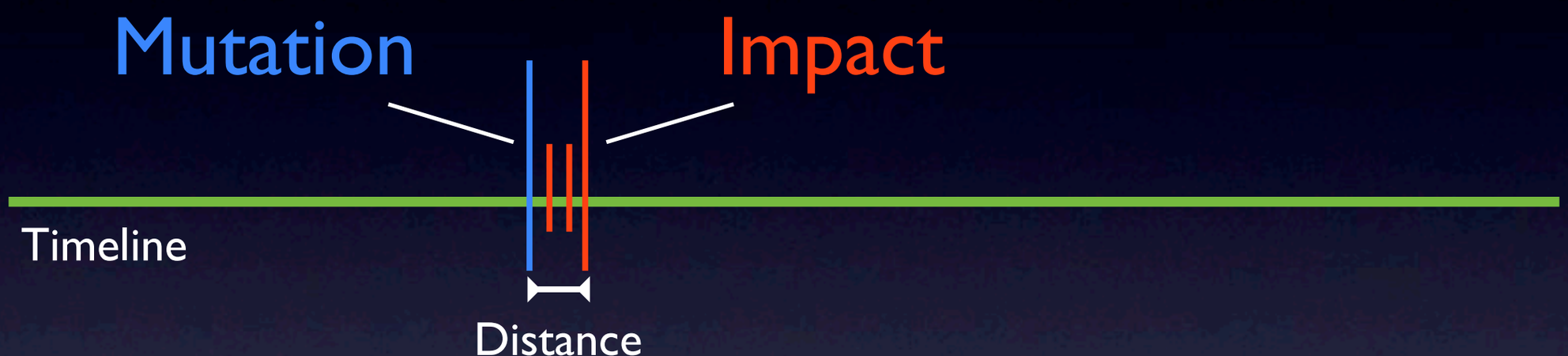
Future Work

- Find other impact measures:
  - Trace *method return values* and *invariants*
    see our paper at ISSTA 2009
  - Count *methods* instead of classes
  - Use some *distance measure*
- Analyze more software projects

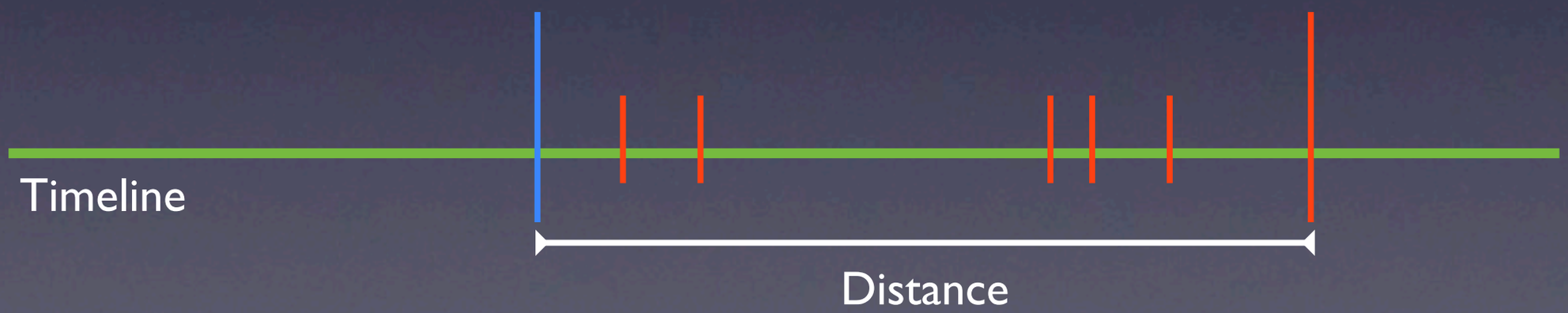
Idea: Distance Measure



Idea: Distance Measure



Early impact ⇒ equivalent



Late impact ⇒ non-equivalent

Conclusion

http://www.st.cs.uni-saarland.de/mutation

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

40% equivalent mutants!

http://www.st.cs.uni-saarland.de/mutation

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

40% equivalent mutants!

Manual Classification

Manual Classification is not efficient!

http://www.st.cs.uni-saarland.de/mutation

Random Mutants

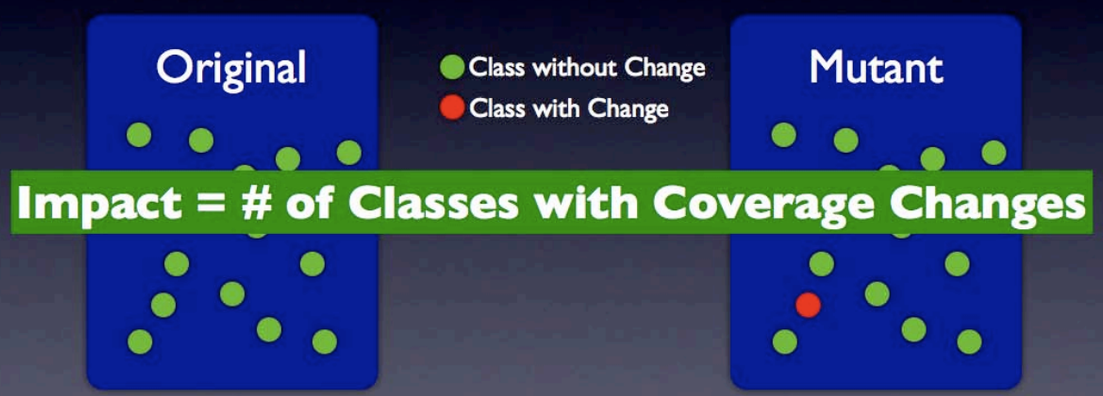
20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

40% equivalent mutants!

Manual Classification

Manual Classification is not efficient!

Coverage Impact



Impact = # of Classes with Coverage Changes

http://www.st.cs.uni-saarland.de/mutation

Random Mutants

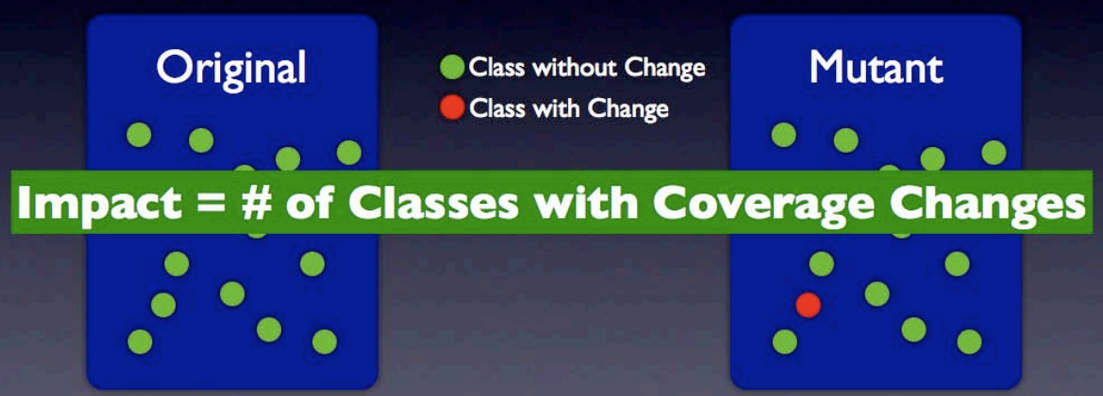
20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% (8)
undecided	10% (2)

40% equivalent mutants!

Manual Classification

Manual Classification is not efficient!

Coverage Impact



Impact = # of Classes with Coverage Changes

Ranking along Impact

✓**Using only top ranked mutations effectively reduces the number of equivalent mutations!**

http://www.st.cs.uni-saarland.de/mutation

Random Mutants

20 randomly chosen mutations from 20 different classes	
non-equivalent	50% (10)
equivalent	40% equivalent
undecided	

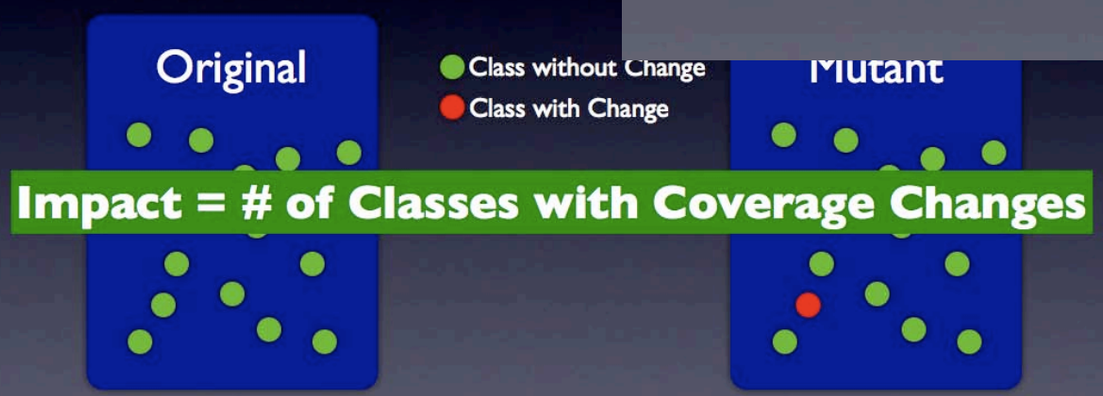
Manual Classification

ation is not efficient!

Future Work

- Find other impact measures.
  - Tracing of method return values.
  - Based on methods instead of classes.
  - Using some distance measure.
- Analyze more software projects.

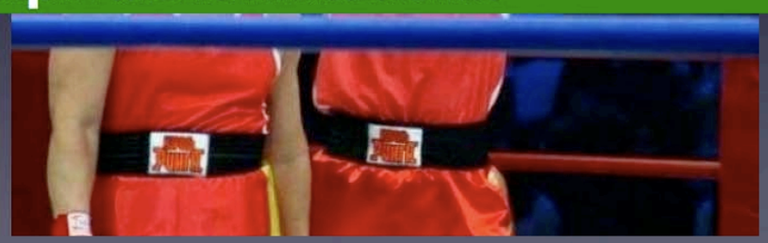
Coverage



Impact = # of Classes with Coverage Changes

along Impact

✓**Using only top ranked mutations effectively reduces the number of equivalent mutations!**



http://www.st.cs.uni-saarland.de/mutation