Are Mutants a Valid Substitute for Real Faults in Software Testing?

•••

Authors: René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, Gordon Fraser

presenter name(s) removed for FERPA considerations
March 23 2017

What is a good test suite?

- **Good test suite** → detects real faults (bugs)
- **How is it measured:** code coverage
- **Problem:** Set of all possible faults unknown
- **Possible Solution:** mutation analysis controlled for code coverage

What is a Mutant?

Mutants: artificial faults (one variation each) that are systematically introduced into the program under test

Original Source Code

```
public int fib( int n) {
   if ((n == 0) || (n == 1))
       return n;
   else
       return fib(n - 1) + fib(n - 2);
   }
```

Mutated Source Code

```
public int fib( int n) {
   if ((n != 0) || (n == 1))
       return n;
   else
       return fib(n - 1) + fib(n - 2);
   }
```

What is Mutation Analysis and Score?

Original Source Code

```
public int fib( int n) {
   if ((n == 0) || (n == 1))
       return n;
   else
       return fib(n - 1) + fib(n - 2);
   }
```

```
@Test // Passes
public void TestFoo() {
     AssertEquals(fib(0) == 0);
}
```

Has no score.

Mutated Source Code

```
public int fib( int n) {
   if ((n != 0) || (n == 1))
       return n;
   else
       return fib(n - 1) + fib(n - 2);
   }
```

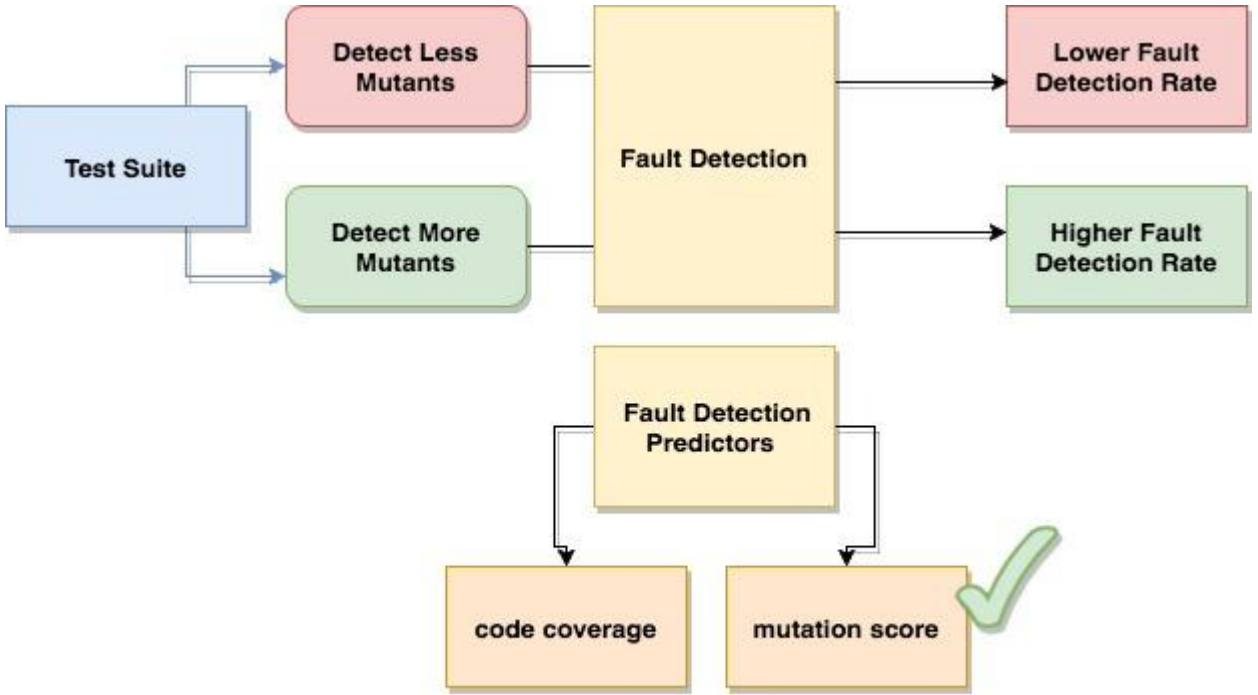
```
@Test // Fails
public void TestFoo() {
     AssertEquals(fib(0) == 0);
}
```

Mutation score 100%. We caught the mutant

Research Questions

1. Are real faults coupled to mutants generated by commonly used mutation operators?
2. What types of real faults are not represented by mutants?
3. Is mutant detection correlated with real fault detection?

Key Idea

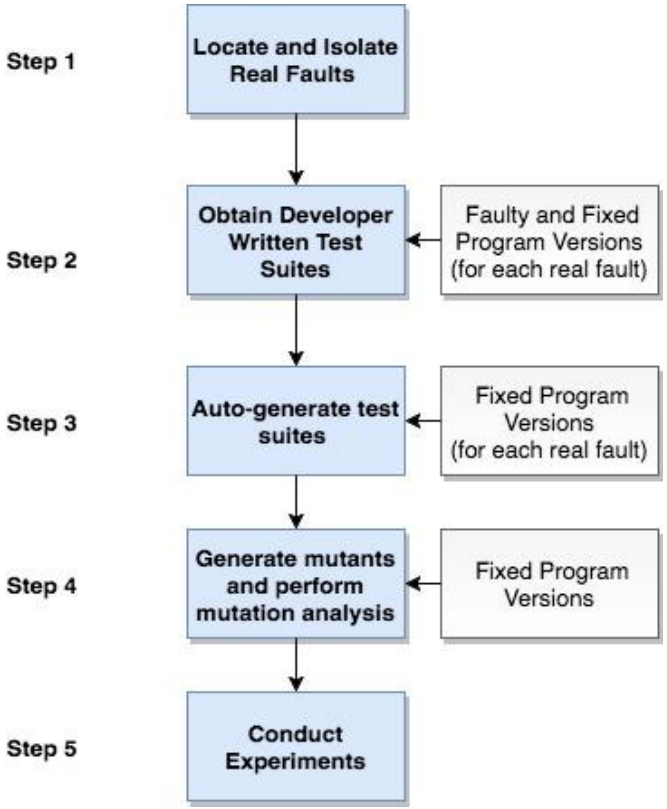


mutation score is a better predictor than code coverage

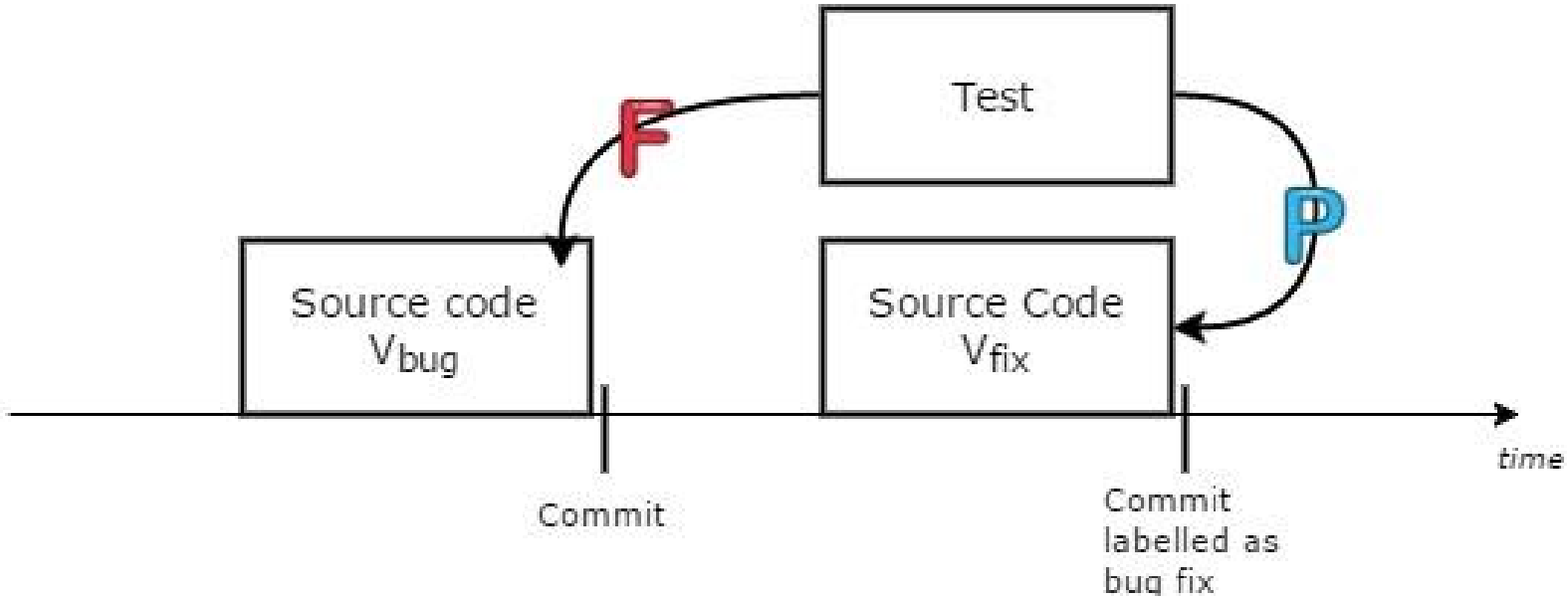
Contributions Made by Paper

- **Develops**
  - Largest study on subject composed of 357 faults, 230K mutants, and test suites
- **Explores**
  - Coupling effect between real faults and mutants
  - Correlation between mutation detection and real fault detection
  - Limitations of mutation analysis

Methodology of Experiment



Step 1: Reproduce and Isolate Real Faults

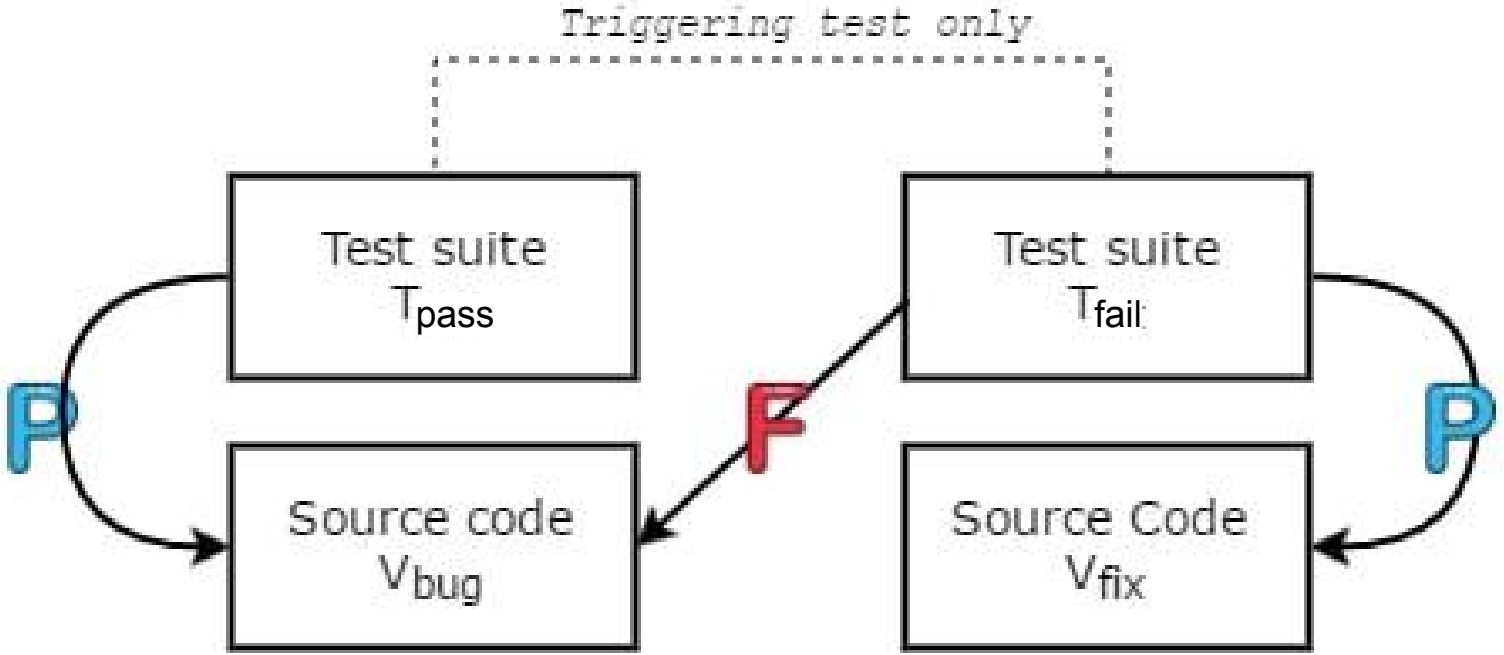


Discard any fault that cannot be reproduced

Reproducible and Isolated Real Faults: Summary

	Candidate revisions	Compilable revisions	Reproducible faults	Isolated faults
Chart	80	62	28	26
Closure	316	227	179	133
Math	435	304	132	106
Time	75	57	29	27
Lang	273	186	69	65
Total	1179	836	437	357

Step 2: Obtain Developer-Written Test Suites



Triggering tests are tests that expose the real fault in V_1 while passing on V_2

Step 3: Automatically Generate Test Suites

Automatically generate test suites using three test generation tools:

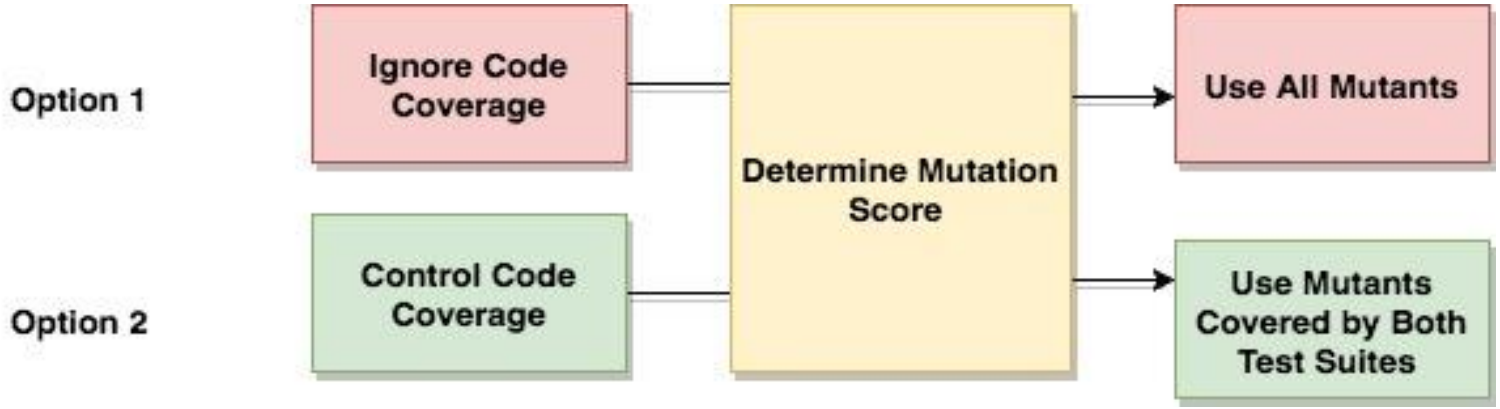
- EvoSuite
  - Branch coverage
  - Weak mutation testing
  - Strong mutation testing
- Randoop
- JCrasher

Then, automatically remove all failing or uncompilable tests.

Step 4: Perform Mutation Analysis

- Major mutation framework
  - Create mutant versions and perform mutation analysis
- Only classes that were modified by the bug fix were mutated
- Major computed mutation coverage and mutation score for each test suite

Step 5: Conduct Experiments



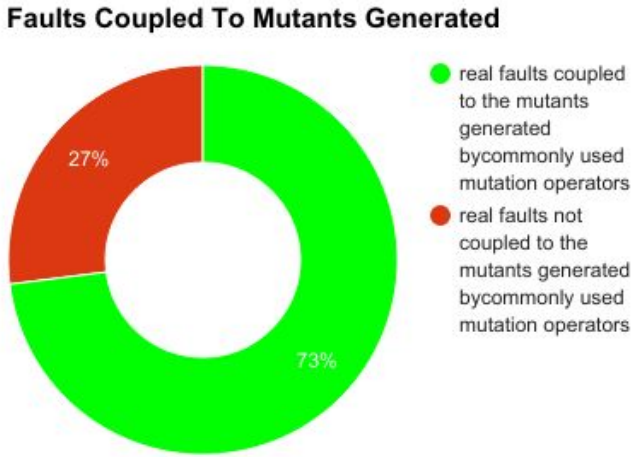
How were the Experiments Analysed?

- Chi-square test to determine significant association between mutants and real faults
- Determined the number of real faults to at least one generated mutant
- Measured the sensitivity of the mutation score to the detection of a single fault

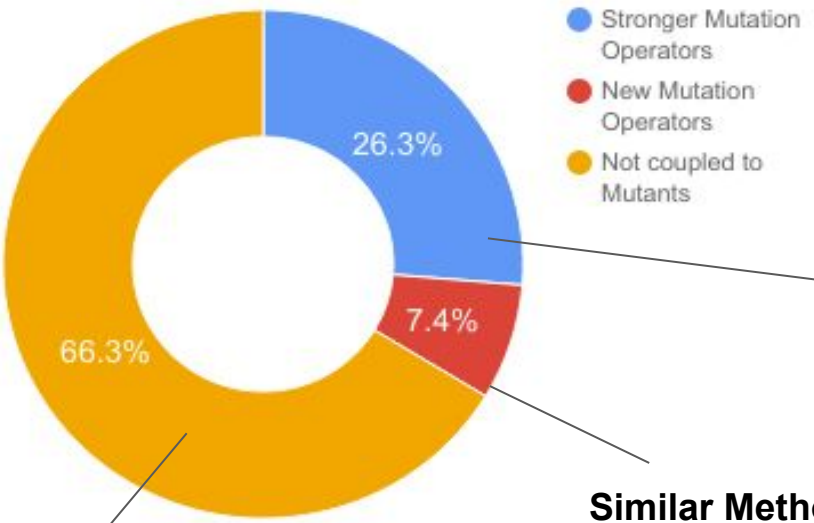
Results

Are Real Faults Coupled to Mutants Generated By Commonly Used Mutation Operators?

- 2 mutants are coupled to a single real fault (on average when controlled for code coverage)
- The following mutations are more often coupled to real faults than other mutants:
  - conditional operator replacement
  - relational operator replacement
  - statement deletion



What types of real faults are not represented by mutants?



Statement Deletion

Similar Method called

Algorithmic Modification and simplification

Is mutant detection correlated with real fault detection?

- Mutation score ≈ real fault detection rate (most of the time)
- Some faults cannot be represented by mutants
- Mutant detection → positively correlated with real fault detection

Conclusions

- **Recall:** Are mutants a valid substitute for real faults in software engineering?
- **Conclusions:**
  - Yes, most of the time, mutants are a valid substitute for real faults in software engineering
  - Some real faults, however, are not represented by mutants
- **Therefore:**
  - Mutants can aid in fixing bugs in code, but will still require human effort

Discussion Questions

Would adding conditional mutant operators (if-else) help strengthen mutation analysis and its relation to real faults?

Do test suite minimization approaches that control for mutation scores retain their real fault detection effectiveness or does it decrease/increase? Why?

Do algorithms used for fault localization and automatic program repair that are evaluated based on mutation scores perform just as well on real faults?

Is the correlation between mutants and real faults the same in low level languages as it is in high level languages such as Java?

Are the 27% of real faults that are not coupled to mutants a part of the real faults that are not coupled to code coverage or do these two approaches find correlation between different real faults?