Are Mutants a Valid Substitute for Real Faults in Software Testing

Just, R., Jalali, D., Inozemtseva, L., Ernst, M. D., Holmes, R., Fraser, G.

slide author names omitted for FERPA compliance
Mutation Analysis

Program

Generate mutants

Test suite

Mutants

Execute Test suite

Mutant detection rate ≈ Real fault detection rate?
Research Questions

➔ Are real faults coupled to mutants generated by commonly used mutation operators?

➔ What types of real faults are not coupled to mutants?

➔ Is mutant detection correlated with real fault detection?
What are Mutants?

➔ Created by systematically injecting small artificial faults into the program being tested.

➔ Using mutation operators - syntactic variations are made (one per mutant).

➔ Proxy measurement for test suite effectiveness - Mutation score
Mutation Operators

1. Replace constants
2. Replace operators.
3. Modify branch conditions.
4. Delete statements.
Replace Constants

```
public float avg(float[] data) {
    float sum = 0;
    for (float num : data) {
        sum += num;
    }
    return sum / data.length;
}
```

```
public float avg(float[] data) {
    float sum = 1;
    for (float num : data) {
        sum += num;
    }
    return sum / data.length;
}
```
Replace Operators

Program

Generate mutants

Mutants

```java
public float avg(float[] data) {
    float sum = 0;
    for (float num : data) {
        sum += num;
    }
    return sum / data.length;
}
```

```java
public float avg(float[] data) {
    float sum = 0;
    for (float num : data) {
        sum += num;
    }
    return sum * data.length;
}
```

*René Just, UW CSE (mutants_real_faults_fse_slides)*
Delete Statements

Program

Generate mutants

Mutants

public float avg(float[] data) {
    float sum = 0;
    for (float num : data) {
        sum += num;
    }
    return sum / data.length;
}

public float avg(float[] data) {
    float sum = 0;
    for (float num : data) {
    }
    return sum / data.length;
}

*René Just, UW CSE (mutants_real_faults_fse_slides)
## Related Work-Summary

<table>
<thead>
<tr>
<th></th>
<th>Real faults</th>
<th>LOC</th>
<th>Tests suites</th>
<th>Mutation operators</th>
<th>Mutants evaluated</th>
<th>Coverage controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>[8]</td>
<td>12</td>
<td>1,000</td>
<td>gen</td>
<td>Rc, Ri, Ro</td>
<td>1%</td>
<td>no</td>
</tr>
<tr>
<td>[1]</td>
<td>38</td>
<td>5,905</td>
<td>gen</td>
<td>Rc, Ro, Nbc, Ds</td>
<td>10%</td>
<td>no</td>
</tr>
<tr>
<td>[27]</td>
<td>38</td>
<td>5,905</td>
<td>gen</td>
<td>Rc, Ri, Ro, Nbc, Ds</td>
<td>10%</td>
<td>no</td>
</tr>
<tr>
<td>Our study</td>
<td>357</td>
<td>321,000</td>
<td>gen/dev</td>
<td>Rc, Ro, Mbc, Ds</td>
<td>100%</td>
<td>yes</td>
</tr>
</tbody>
</table>

Rc - Replace constants, Ri - Replace Identifiers, Ro - Replace operators, Nbc - Negate branch conditions, Ds - Delete statements, Mbc - Modify branch conditions.

Key Idea - Methodology

- Developer written
- Test suite
- Automated
- Real fault
- Real fault detection rate
- Mutant detection rate
- Mutants
- Compare results
Key Idea - Methodology

- Developer written
- Test suite
- Automated
- Real fault
  - Real fault detection rate
- Mutant detection rate
- Mutants
- Compare results
Methodology - Reproducible and isolated real faults

<table>
<thead>
<tr>
<th></th>
<th>Candidate revisions</th>
<th>Compilable revisions</th>
<th>Reproducible faults</th>
<th>Isolated faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chart</td>
<td>80</td>
<td>62</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>Closure</td>
<td>316</td>
<td>227</td>
<td>179</td>
<td>133</td>
</tr>
<tr>
<td>Math</td>
<td>435</td>
<td>304</td>
<td>132</td>
<td>106</td>
</tr>
<tr>
<td>Time</td>
<td>75</td>
<td>57</td>
<td>29</td>
<td>27</td>
</tr>
<tr>
<td>Lang</td>
<td>273</td>
<td>186</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1179</strong></td>
<td><strong>836</strong></td>
<td><strong>437</strong></td>
<td><strong>357</strong></td>
</tr>
</tbody>
</table>

![Diagram showing the relationship between source code diff, features & refactorings, bug fix, and final version.](image)
Key Idea - Methodology

Developer written → Test suite → Automated

Real fault → Real fault detection rate

Mutant detection rate → Mutants

Compare results
Methodology- Mutant Generation

➔ 230,000 mutants generated using Major mutation framework.

➔ Mutation operators as discussed before.
Key Idea - Methodology

- Developer written
- Test suite
- Automated
- Real fault
- Real fault detection rate
- Mutant detection rate
- Mutants
- Compare results
Methodology- Test suite

Developer written test suite

→ Test pair $\langle T_{\text{bug}} \text{ and } T_{\text{fix}} \rangle$

→ Average statement coverage of $T_{\text{bug}}$ : 90%

Automatically generated test suite

→ Generated using EvoSuite, Randoop and JCrasher.

→ Around 35000 test suites.

→ Average statement coverage : 55%
Developers written test suites:

**Figure 2:** Relationship between the $i$-th obtained test suite pair $\langle T_{pass}^i, T_{fail}^i \rangle$ and the developer-written test suites $T_{bug}$ and $T_{fix}$.
Evaluation

RQ1: Are real faults coupled to mutants generated by using mutation operators?

→ Test pair $< T_{\text{bug}}$ and $T_{\text{fix}} > : T_{\text{fix}}$ (mutant detection rate) $> T_{\text{bug}}$

→ Results: Mutant detection rate increased for 73% of faults.

→ Conditional operator replacement, Relational operator replacement, and statement deletion mutants.
Evaluation

RQ2: Type of faults not represented by mutants?

➔ Qualitative study of 27% of the faults.

➔ Weakness or general limitation.
Evaluation

RQ2: Type of faults not represented by mutants?

➔ Qualitative study of 27% of the faults.

➔ Weakness or general limitation.
Weak or missing mutation operator

Examples

➔ Argument Omission

- return solve(min, max);
+ return solve(f, min, max);

➔ Statement Deletion

}  
+ return false;  
}  
case 4: {  
char ch = str.charAt(0);
Real faults not coupled to Mutants

Examples

➔ Code Deletion
   
   ```java
   if (childType.isDict()) {
       ...
   - } else if (n.getJSType != null &&
   -       parent.isAssign()) {
   -       return;
   } ...
   ```

➔ Similar method calls

   ```java
   - return getPct((Comparable<?>) v);
   + return getCumPct((Comparable<?>) v);
   ```
Evaluation

RQ3: Is mutant detection correlated with fault detection?
Contributions

1. 357 new developer fixed and manually-verified real faults with test suites.

2. Most comprehensive study to date on mutation testing.

3. Investigation confirmed 73% real faults coupled with mutants.

4. Concrete suggestions for improving mutation analysis and identifying its inherent limitations.

5. Significant correlation between mutant detection and fault detection.
Discussion

➔ Are the results representative of the software projects since only 5 projects are under consideration?

➔ Do the results apply to other programming languages as well?

➔ Does the removal of faults introduce a fault bias?

➔ Can we minimize the test suite based on the mutation scores?

➔ Can we generate a test suite based on mutants?
Thank you 😊