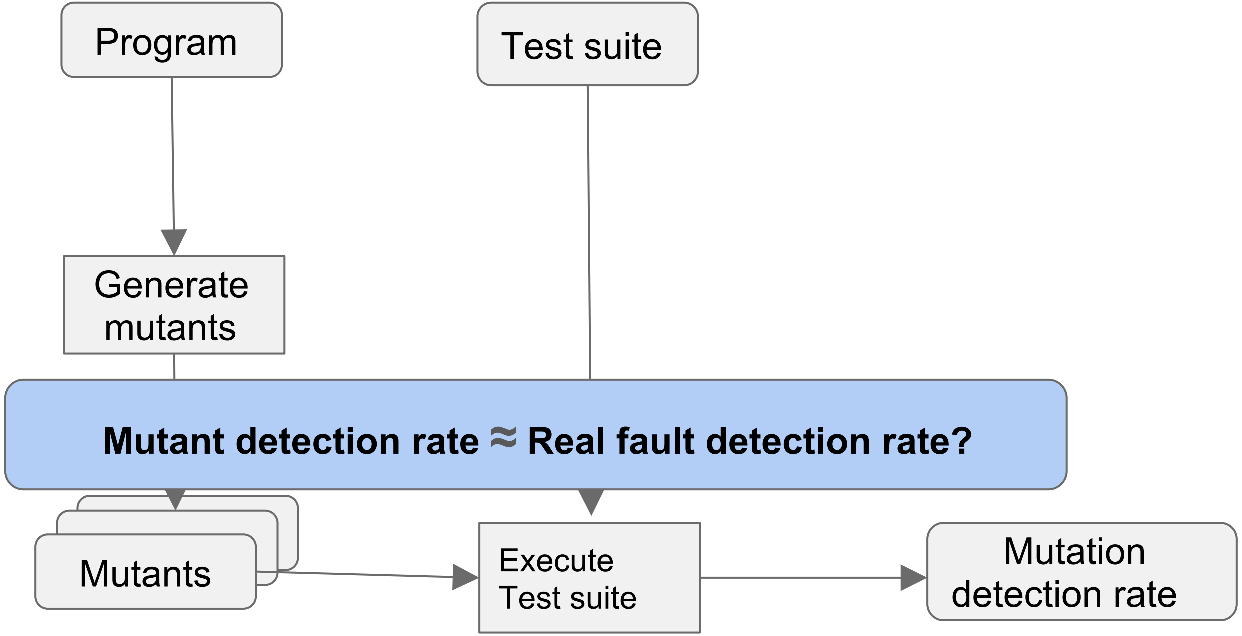
Are Mutants a Valid Substitute for Real Faults in Software Testing

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

slide author names omitted for FERPA compliance

Mutation Analysis



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

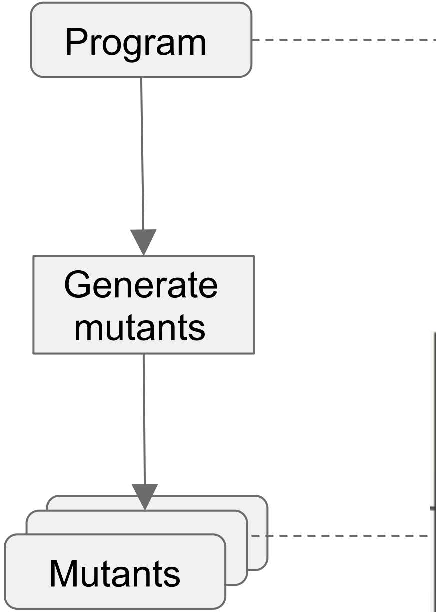
1. Replace constants

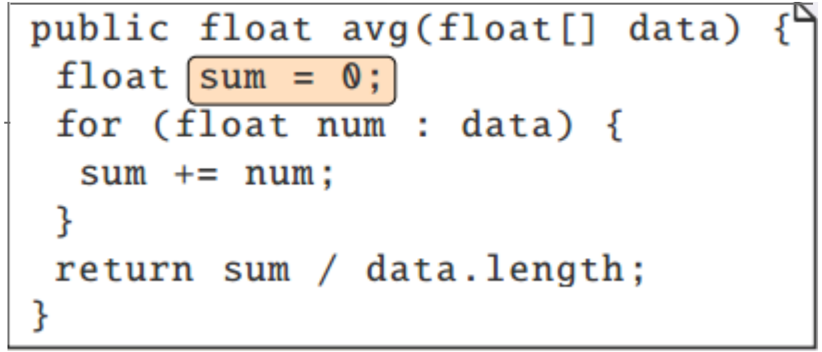
2. Replace operators.

3. Modify branch conditions.

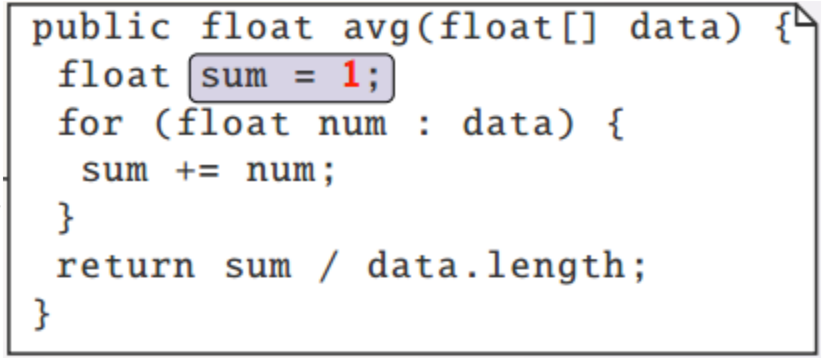
4. Delete statements.

Replace Constants





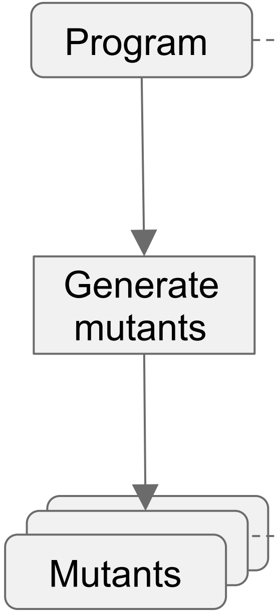
```
public float avg(float[] data) {
 float sum = 0;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```

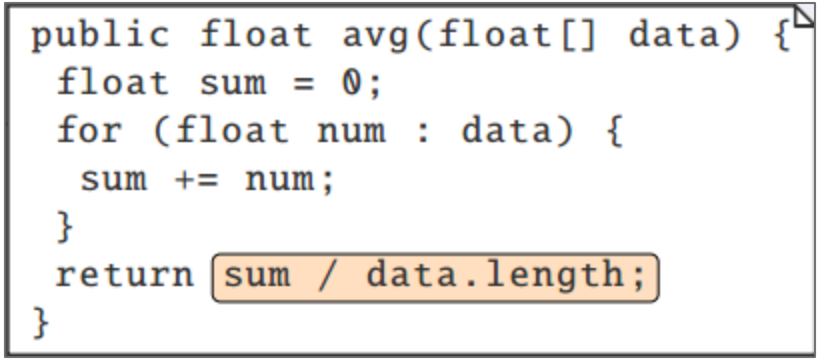


```
public float avg(float[] data) {
 float sum = 1;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```

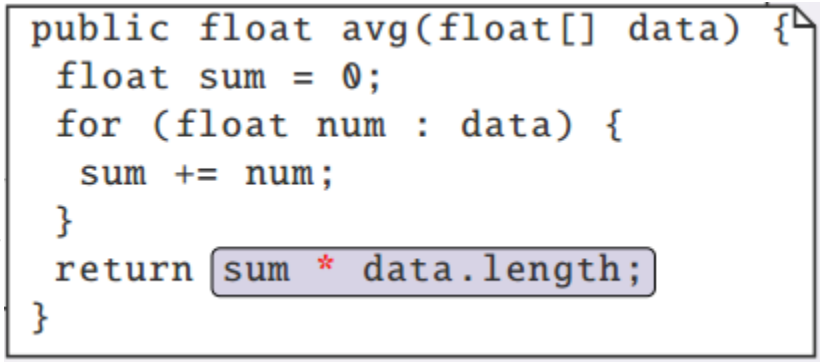
*René Just, UW CSE (mutants_real_faults_fse_slides)

Replace Operators





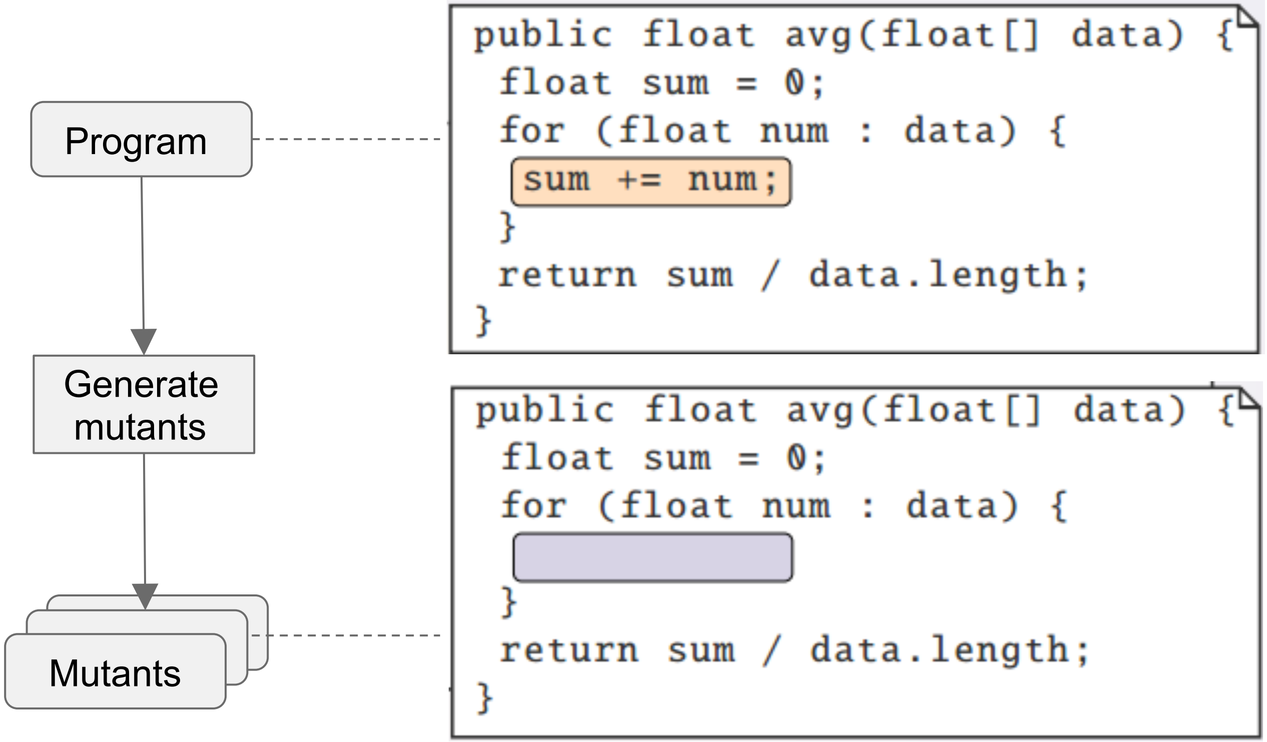
```
public float avg(float[] data) {
 float sum = 0;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```



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

*René Just, UW CSE (mutants_real_faults_fse_slides)

Delete Statements



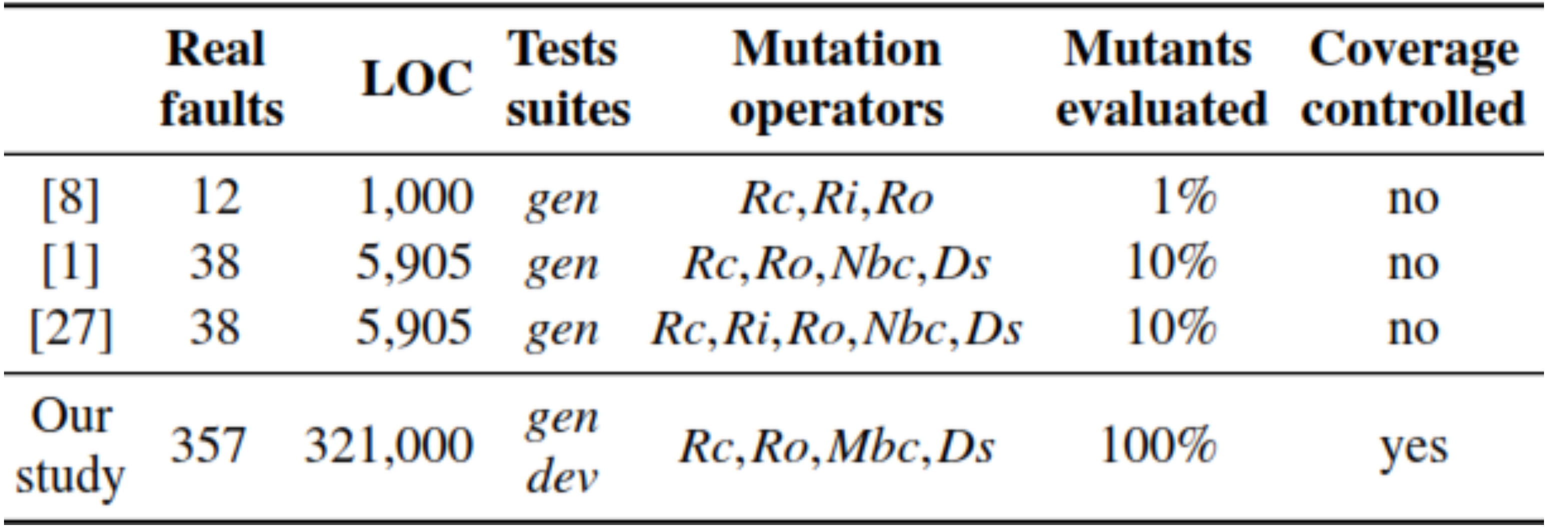
```
public float avg(float[] data) {
 float sum = 0;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```

```
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

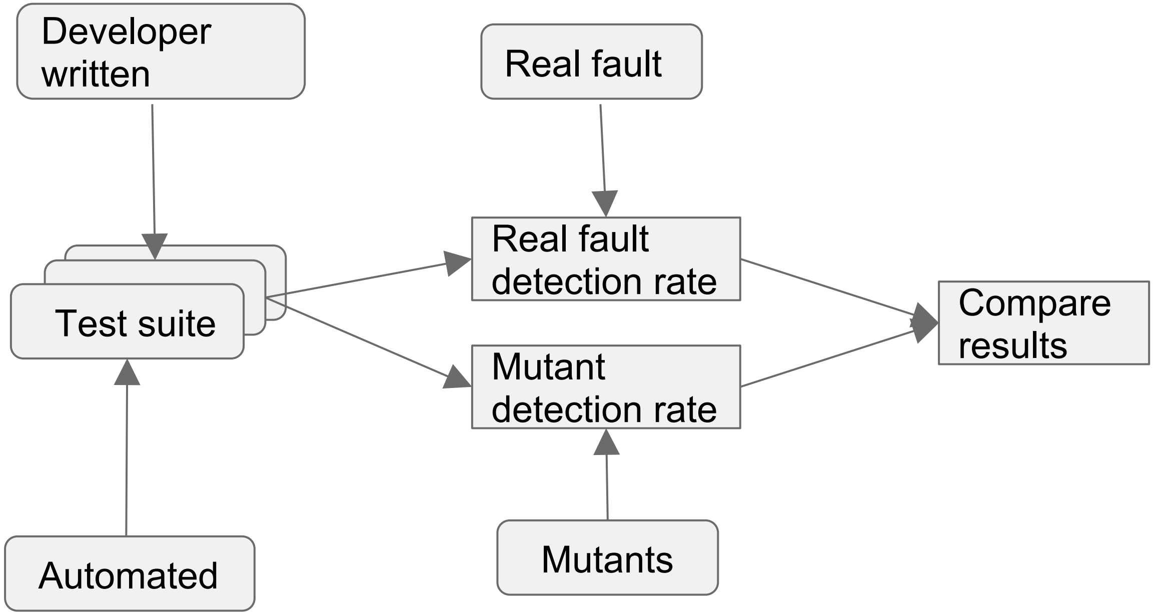


	Real faults	LOC	Tests suites	Mutation operators	Mutants evaluated	Coverage controlled
[8]	12	1,000	*gen*	*Rc,Ri,Ro*	1%	no
[1]	38	5,905	*gen*	*Rc,Ro,Nbc,Ds*	10%	no
[27]	38	5,905	*gen*	*Rc,Ri,Ro,Nbc,Ds*	10%	no
Our study	357	321,000	*gen* *dev*	*Rc,Ro,Mbc,Ds*	100%	yes

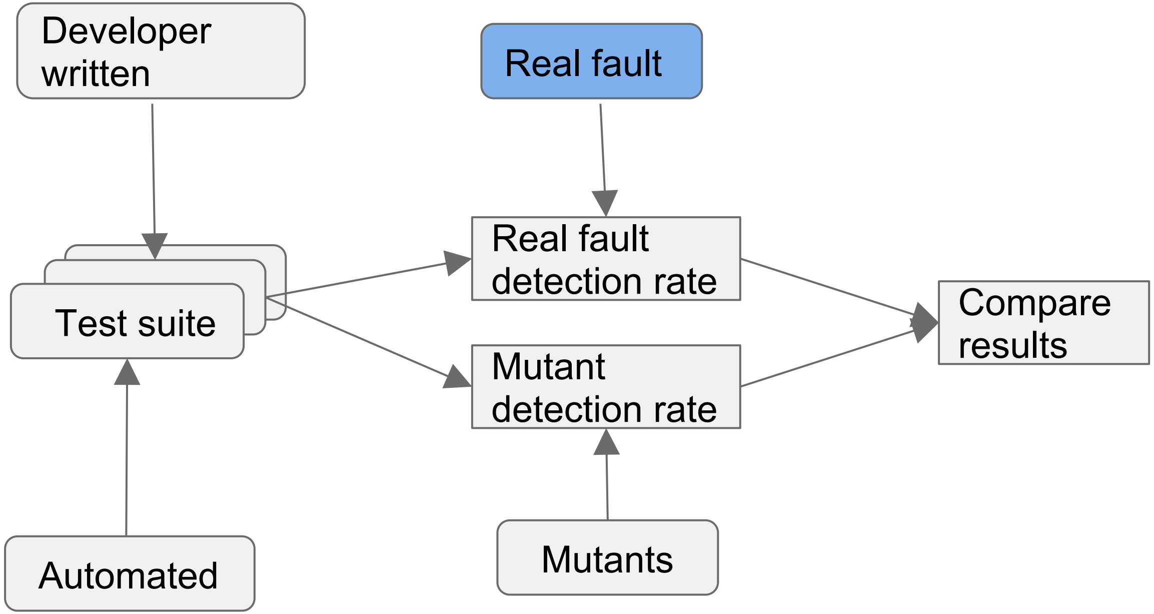
Rc - Replace constants, Ri - Replace Identifiers, Ro - Replace operators, Nbc - Negate branch conditions,
Ds - Delete statements, Mbc - Modify branch conditions.

[8] : M. Daran and P. Thévenod-Fosse. Software error analysis: A real case study involving real faults and mutations.
[1] : J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments?
[27] : A. S. Namin and S. Kakarla. The use of mutation in testing experiments and its sensitivity to external threats.

Key Idea - Methodology



Key Idea - Methodology



Developer written

Real fault

Test suite

Real fault detection rate

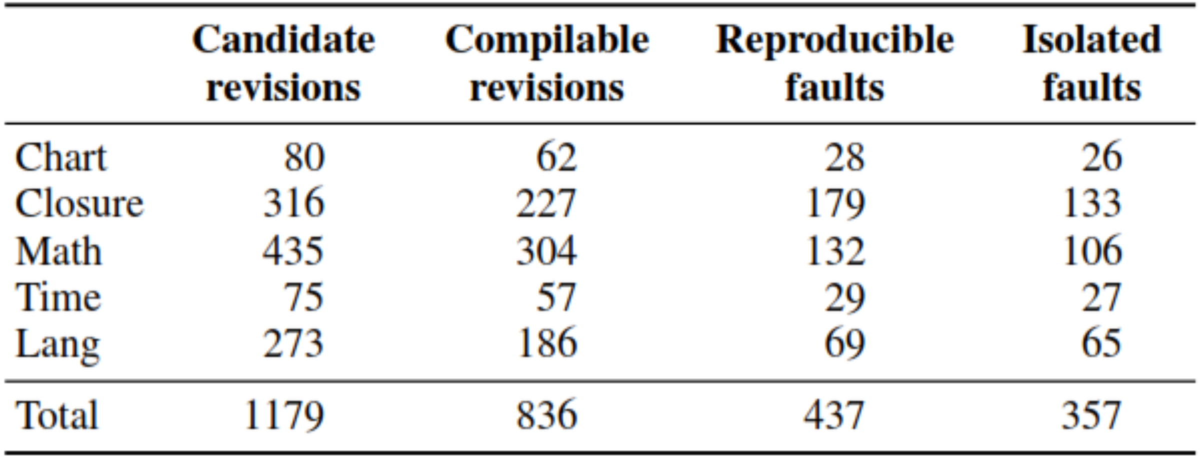
Mutant detection rate

Compare results

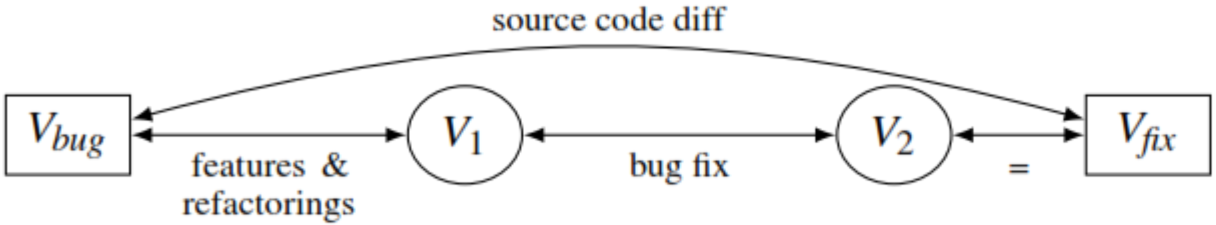
Automated

Mutants

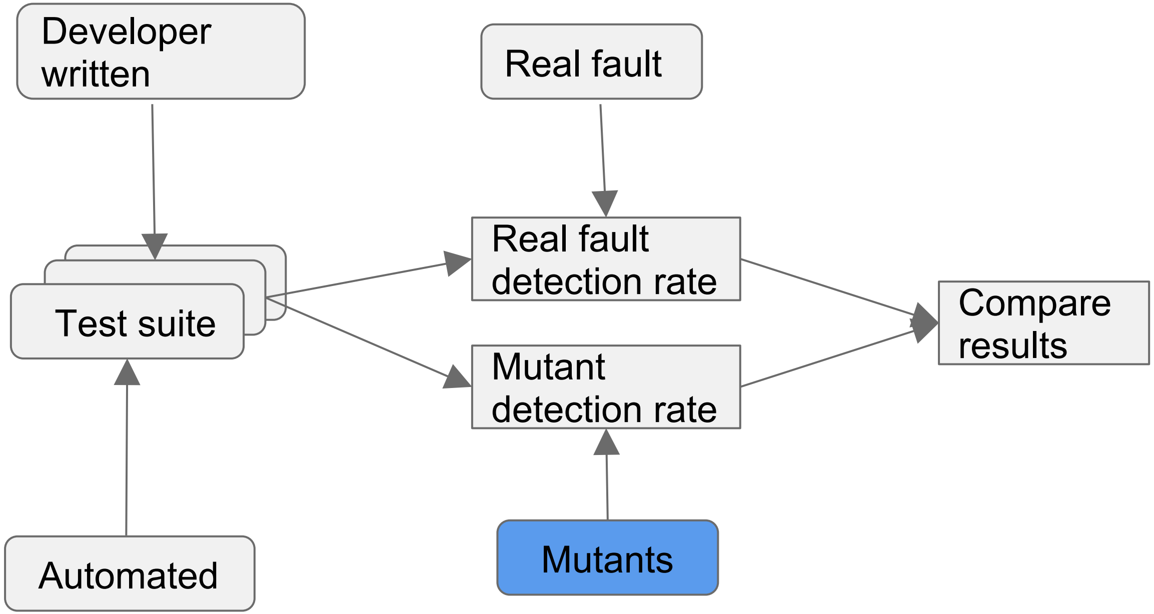
Methodology - Reproducible and isolated real faults



	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



Key Idea - Methodology



Developer written

Test suite

Automated

Real fault

Real fault detection rate

Mutant detection rate

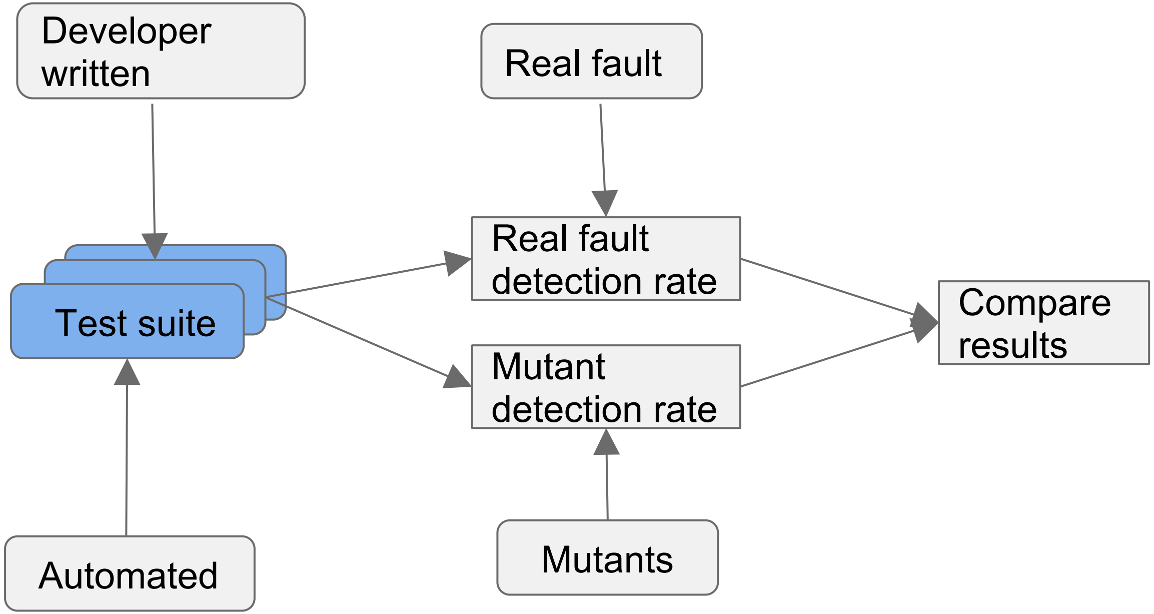
Mutants

Compare results

Methodology- Mutant Generation

- 230,000 mutants generated using Major mutation framework.
- Mutation operators as discussed before.

Key Idea - Methodology



Developer written

Real fault

Test suite

Real fault detection rate

Mutant detection rate

Compare results

Automated

Mutants

Methodology- Test suite

Developer written test suite

- ➔ Test pair < T_{bug} and T_{fix} >
- ➔ Average statement coverage of T_{bug} : 90%

Automatically generated test suite

- ➔ Generated using EvoSuite, Randoop and JCrasher.
- ➔ Around 35000 test suites.
- ➔ Average statement coverage : 55%

Developer written test suites:

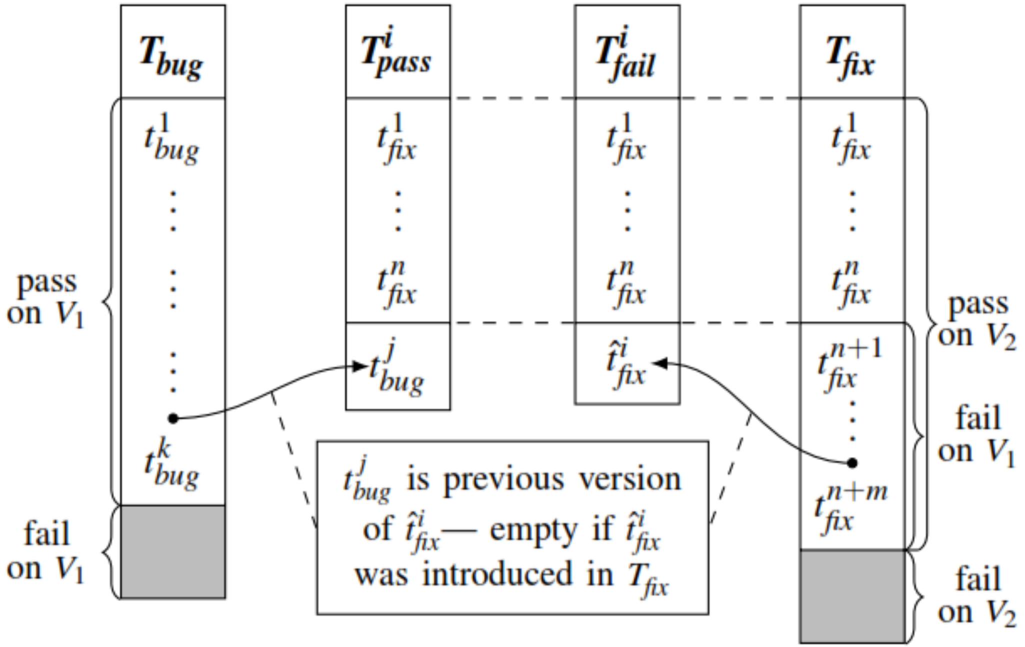


Figure 2: Relationship between the *i*-th obtained test suite pair $\langle T^i_{pass}, T^i_{fail} \rangle$ and the developer-written test suites T_{bug} and T_{fix}.

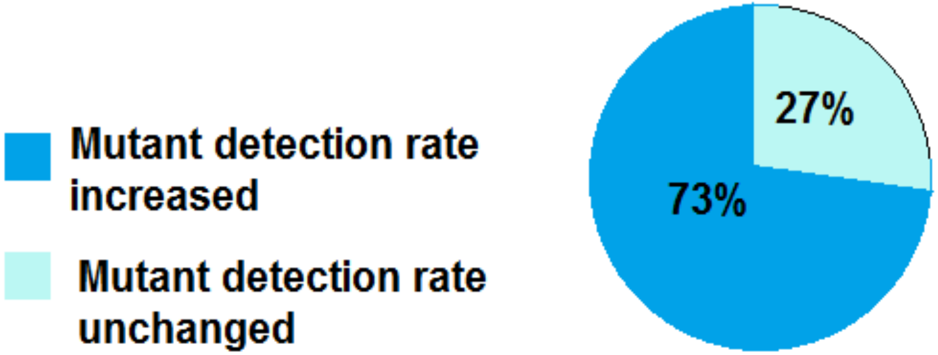
Evaluation

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

➔ Test pair < T_{bug} and T_{fix} > : T_{fix} (mutant detection rate) > T_{bug}

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

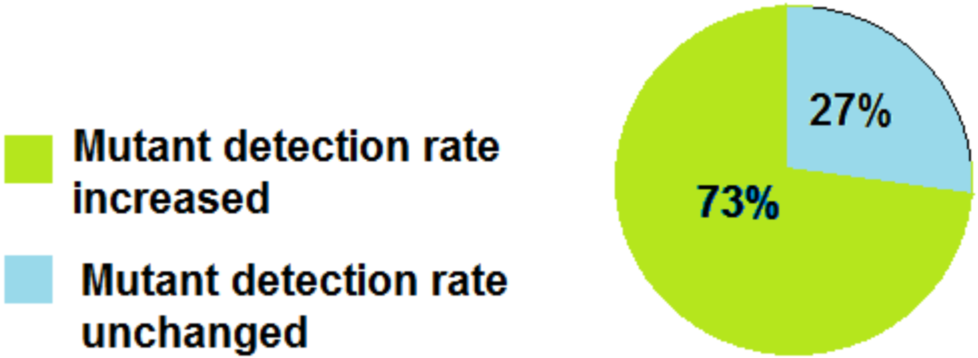
➔ Conditional operator replacement, Relational operator replacement, and statement deletion mutants.



Evaluation

RQ2: Type of faults not represented by mutants?

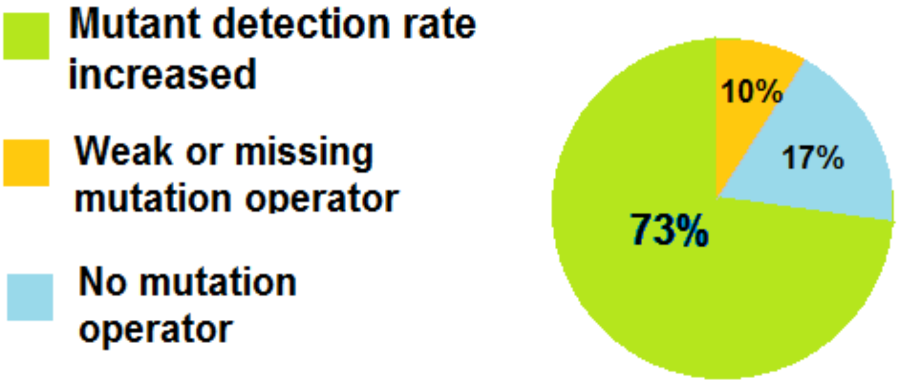
- ➔Qualitative study of 27% of the faults.
- ➔ Weakness or general limitation.



Evaluation

RQ2: Type of faults not represented by mutants?

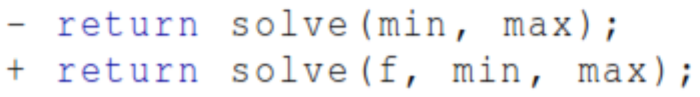
- ➔ Qualitative study of 27% of the faults.
- ➔ Weakness or general limitation.



Weak or missing mutation operator

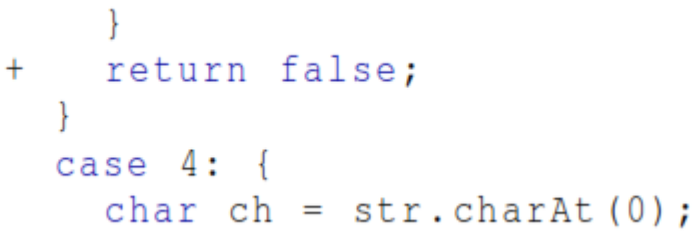
Examples

- ➔ Argument Omission



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

- ➔ Statement Deletion

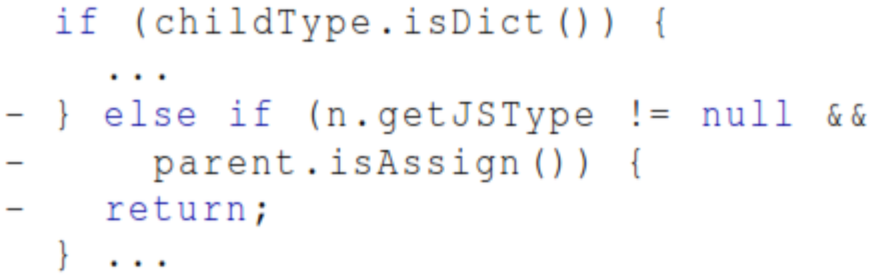


```
    }
+   return false;
  }
  case 4: {
    char ch = str.charAt(0);
```

Real faults not coupled to Mutants

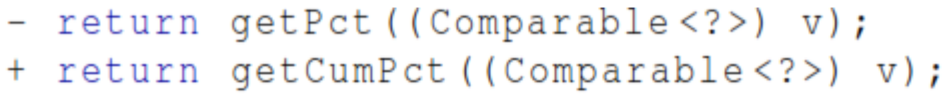
Examples

- ➔ Code Deletion



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

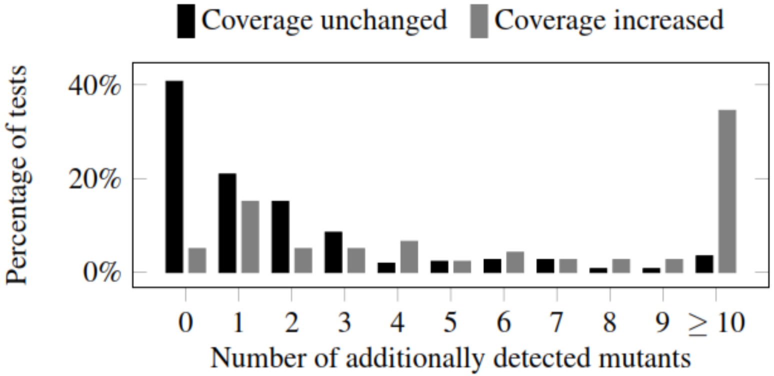
- ➔ Similar method calls

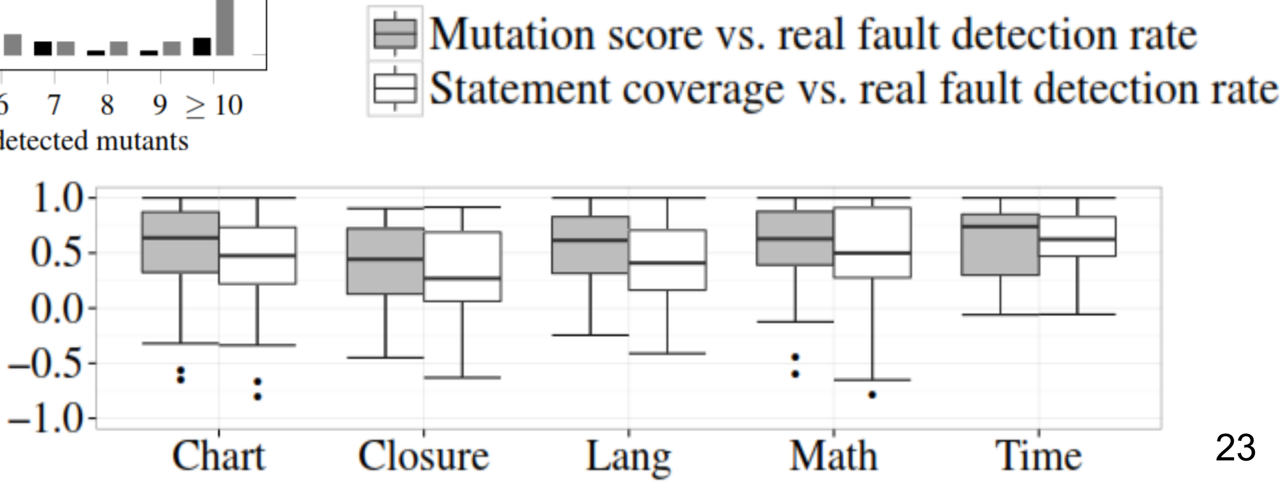


```
- return getPct((Comparable<?>) v);
+ return getCumPct((Comparable<?>) v);
```

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 ☺