Using Controlled Numbers of Real Faults and Mutants to Empirically Evaluate Coverage-Based Test Case Prioritization

David Paterson
University of Sheffield

Gregory Kapfhammer
Allegheny College

Gordon Fraser
University of Passau

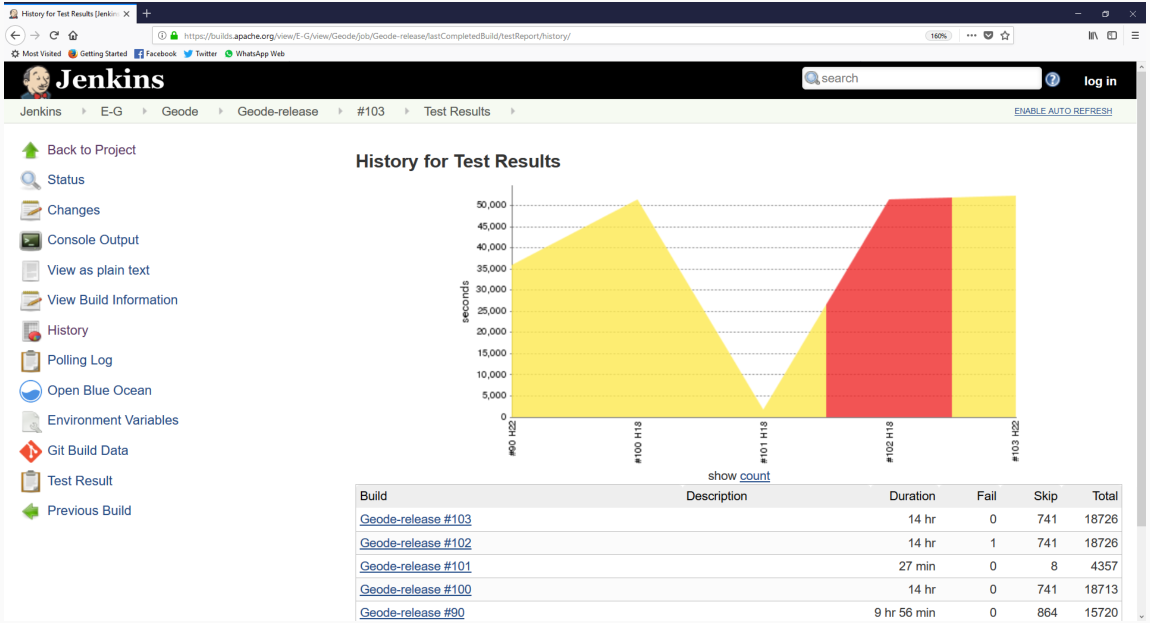
Phil McMinn
University of Sheffield

Workshop on Automation of Software Test
29th May 2018

dpaterson1@sheffield.ac.uk

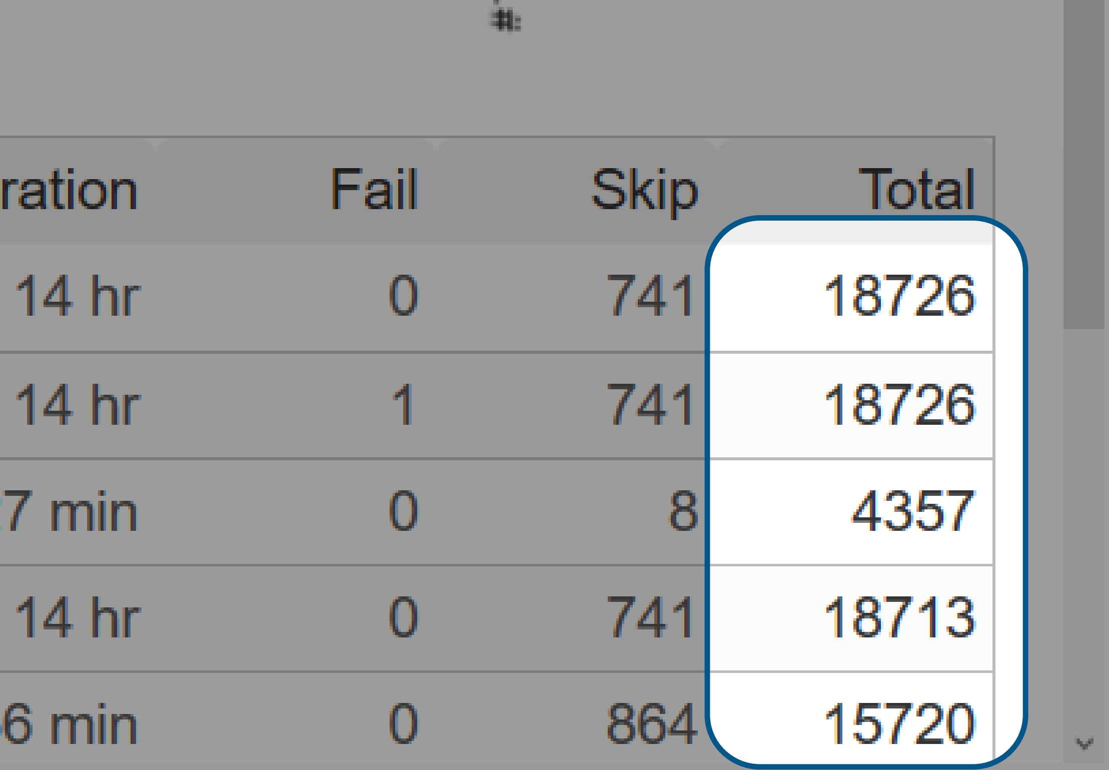
Test Case Prioritization

- Testing is required to ensure the correct functionality of software
- Larger software → more tests → longer running test suites

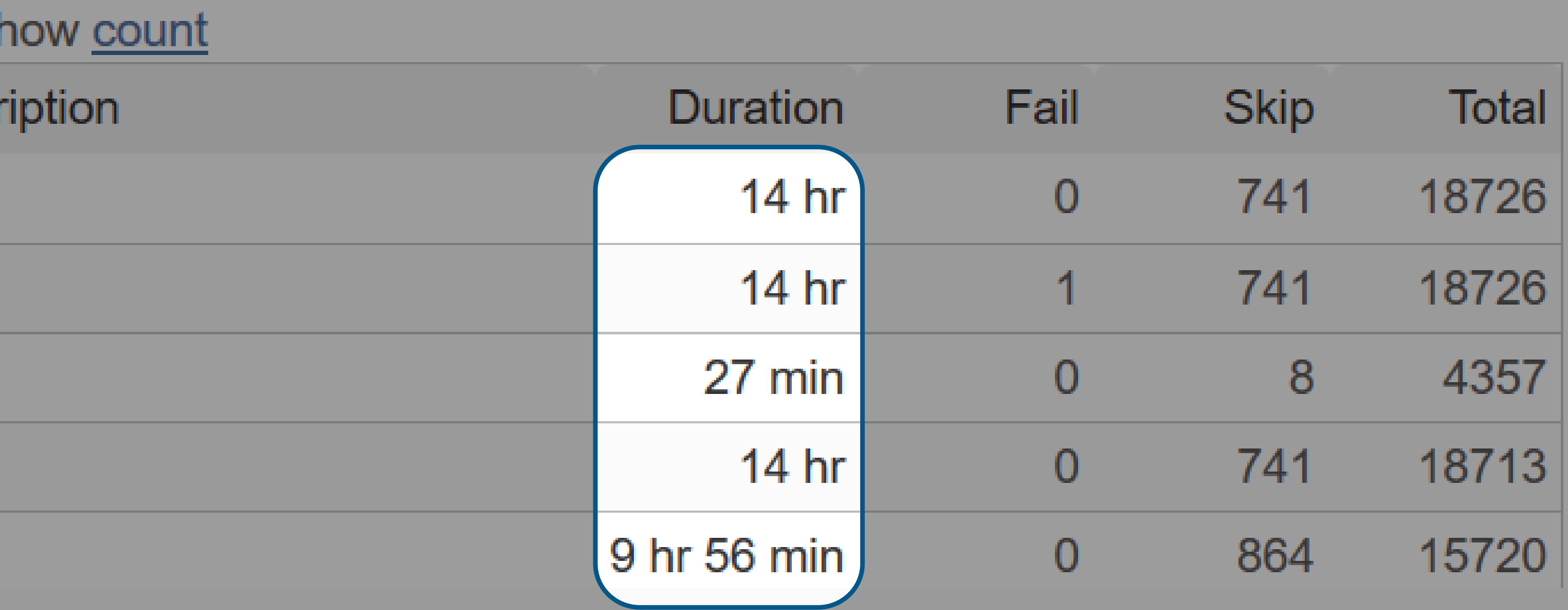


History for Test Results

Build	Description	Duration	Fail	Skip	Total
Geode-release #103		14 hr	0	741	18726
Geode-release #102		14 hr	1	741	18726
Geode-release #101		27 min	0	8	4357
Geode-release #100		14 hr	0	741	18713
Geode-release #90		9 hr 56 min	0	864	15720



ration	Fail	Skip	Total
14 hr	0	741	18726
14 hr	1	741	18726
7 min	0	8	4357
14 hr	0	741	18713
6 min	0	864	15720

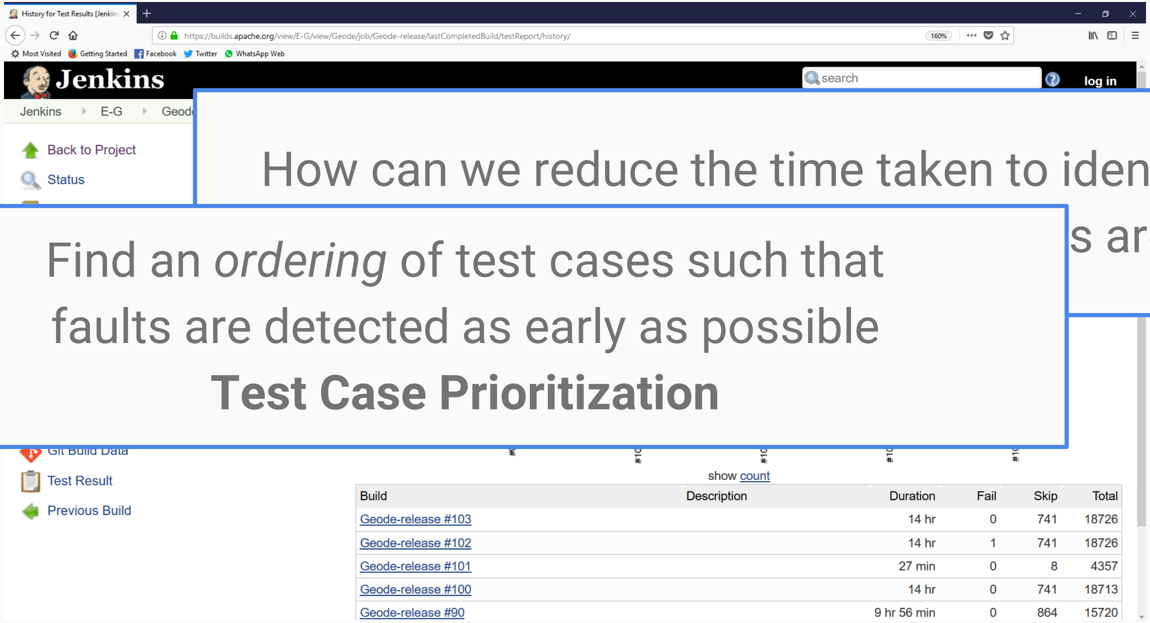


how count

ription	Duration	Fail	Skip	Total
	14 hr	0	741	18726
	14 hr	1	741	18726
	27 min	0	8	4357
	14 hr	0	741	18713
	9 hr 56 min	0	864	15720

Test Case Prioritization

- Testing is required to ensure the correct functionality of software
- Larger software -> more tests -> longer running test suites



History for Test Results [Jenkins

https://builds.apache.org/view/E-G/view/Geode/job/Geode-release/lastCompletedBuild/testReport/history/

Build	Description	Duration	Fail	Skip	Total
Geode-release #103		14 hr	0	741	18726
Geode-release #102		14 hr	1	741	18726
Geode-release #101		27 min	0	8	4357
Geode-release #100		14 hr	0	741	18713
Geode-release #90		9 hr 56 min	0	864	15720

How can we reduce the time taken to identify new

s are found?

Find an *ordering* of test cases such that faults are detected as early as possible

Test Case Prioritization

Types of Fault

Real

Artificial

Test Case Prioritization

Strategy A

- 100 subjects
- Evaluated on **mutants**
- **Score = 0.75**

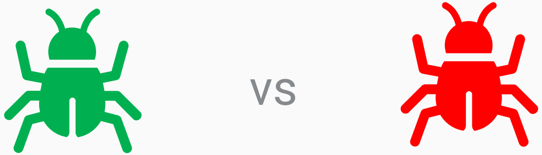
Strategy B

- 100 subjects
- Evaluated on **real faults**
- **Score = 0.72**

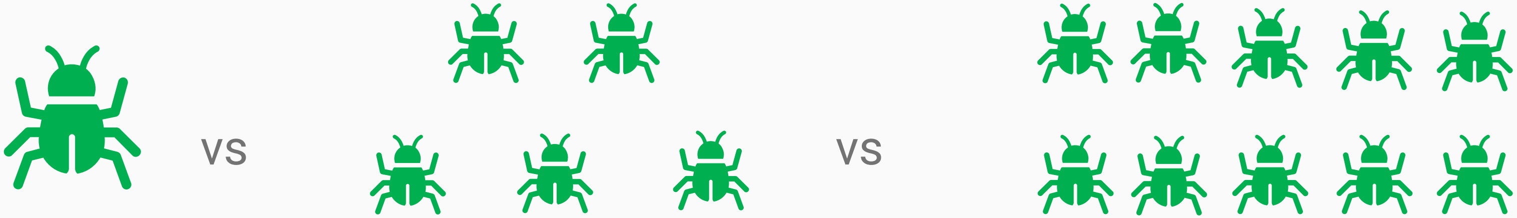
Which strategy performs the best?

Research Objectives

1. Compare prioritization strategies across fault types



2. Investigate the impact of multiple faults



Evaluating Test Prioritization

Average Percentage of Faults Detected (APFD)

- % Faults Found vs % Test Suite executed

- $APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{2n}$

Evaluating Test Prioritization

Average Percentage of Faults Detected (APFD)

- % Faults Found vs % Test Suite executed

- $APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{2n}$

uating Test Prioritization

rage Percentage of Faults Detected (APFD)

% Faults Found vs % Test Suite executed

$$APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{2n}$$

ing Test Prioritization

e Percentage of Faults Detected (APFD)

aults Found vs % Test Suite executed

$$PFD = 1 - \frac{\sum_{i=1}^{m} \boxed{TF_i}}{mn} + \frac{1}{2n}$$

erage Percentage of Faults Detected (APFD)

% Faults Found vs % Test Suite executed

$$APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{\boxed{mn}} + \frac{1}{2n}$$

e of Faults Detected (APFD)

vs % Test Suite executed

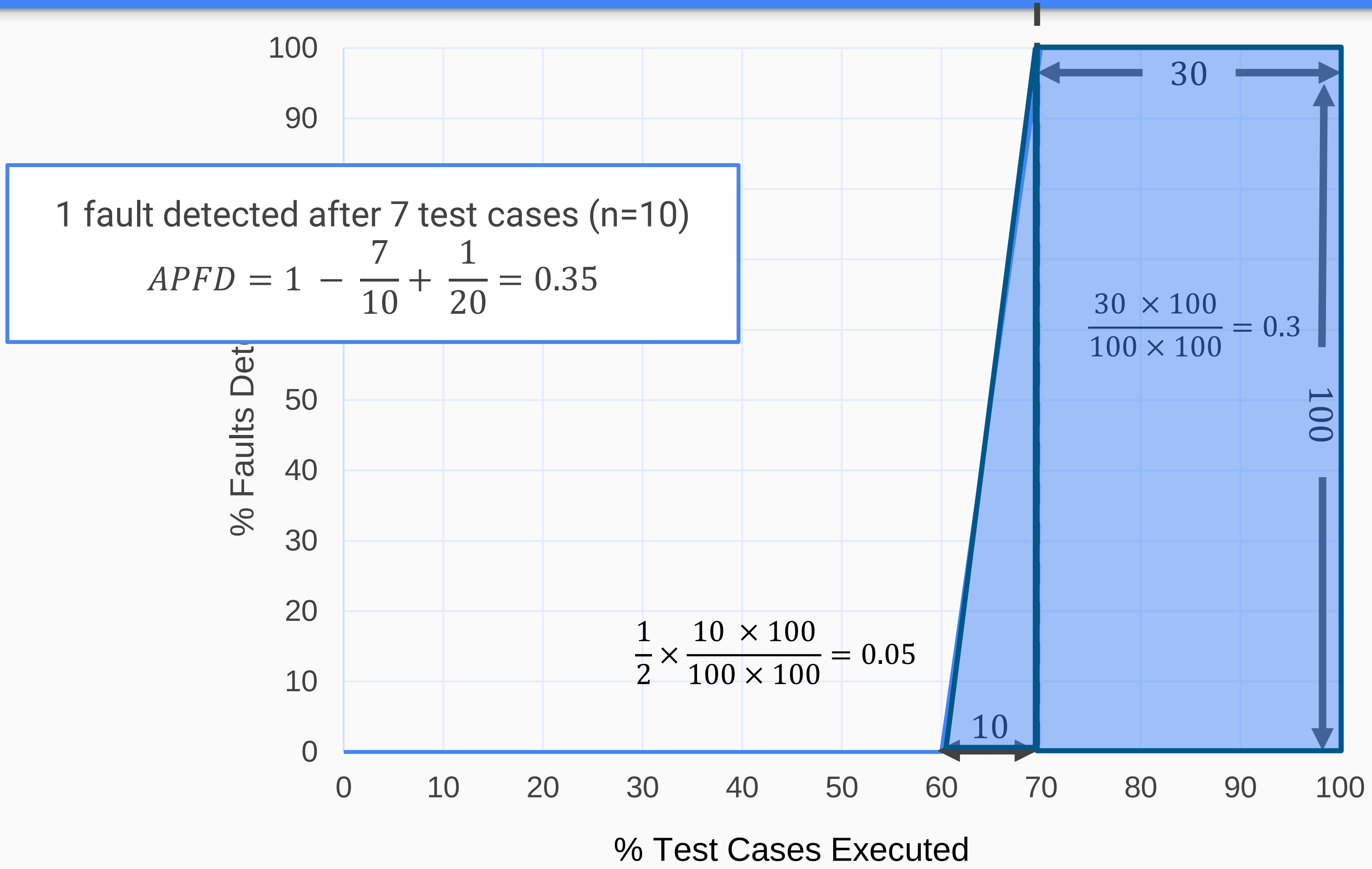
$$= 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{\boxed{2n}}$$

Evaluating Test Prioritization

Average Percentage of Faults Detected (APFD)

- % Faults Found vs % Test Suite executed
- $APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{mn} + \frac{1}{2n}$
- TCP aims to **maximize** APFD by **minimizing** TF_i

Evaluating Test Prioritization



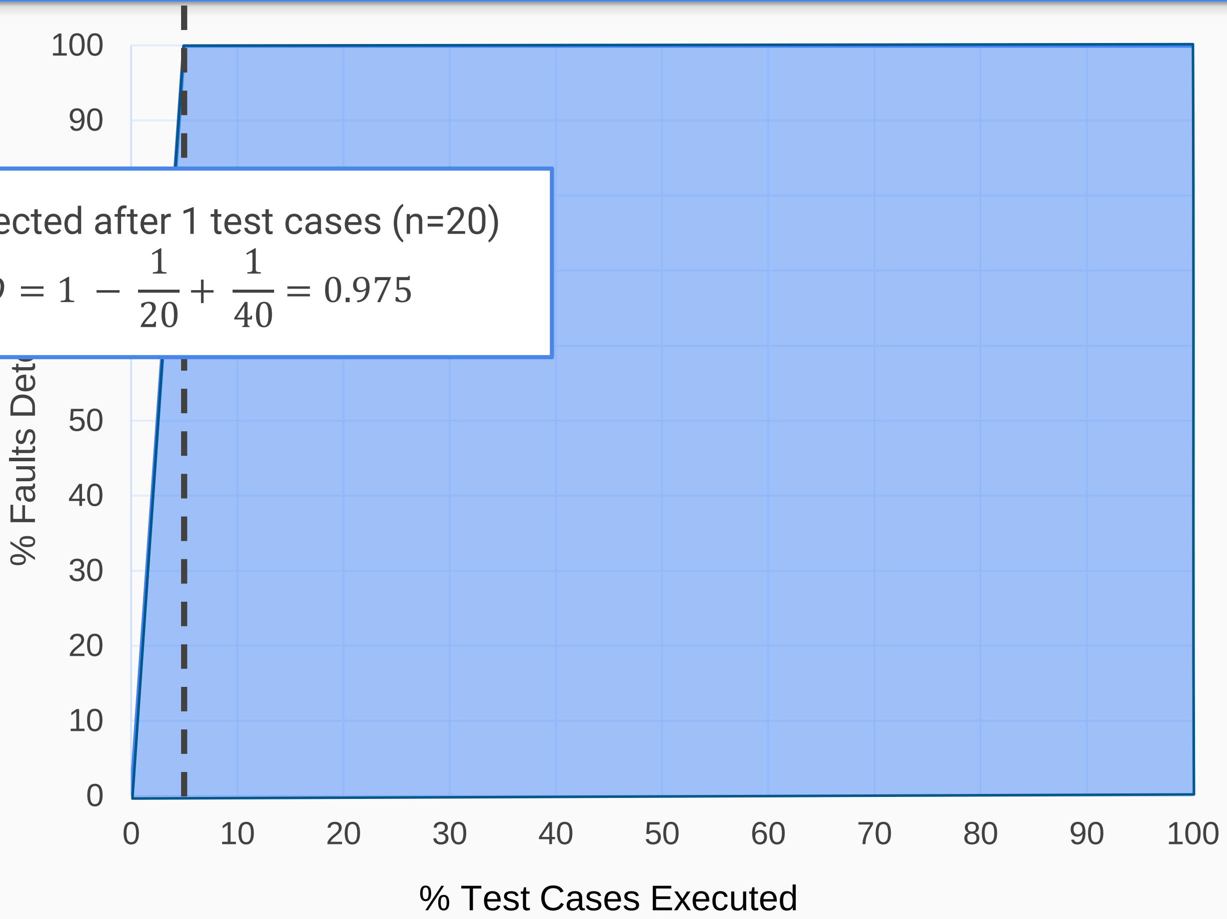
1 fault detected after 7 test cases (n=10)

$$APFD = 1 - \frac{7}{10} + \frac{1}{20} = 0.35$$

$$\frac{30 \times 100}{100 \times 100} = 0.3$$

$$\frac{1}{2} \times \frac{10 \times 100}{100 \times 100} = 0.05$$

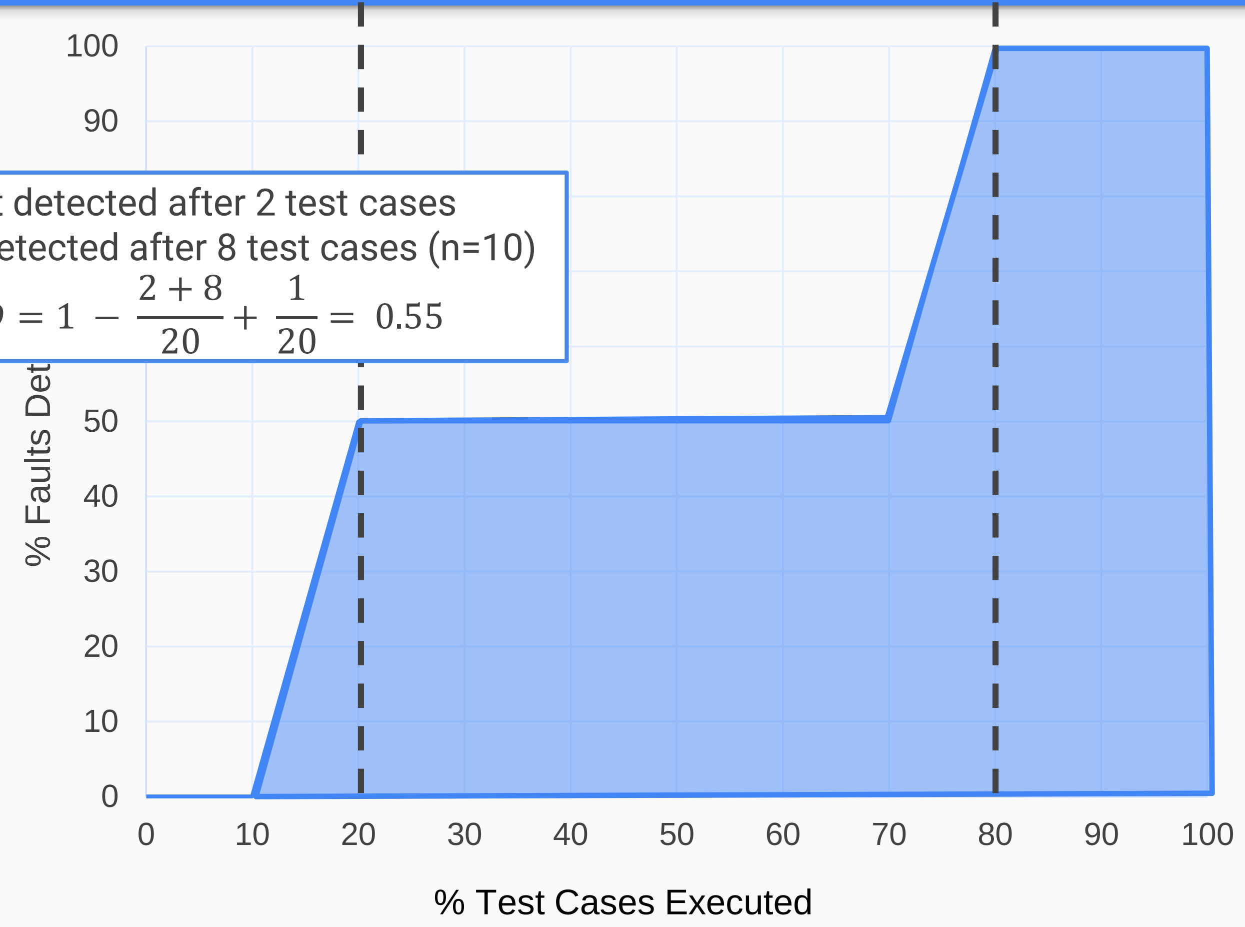
Evaluating Test Prioritization



1 fault detected after 1 test cases (n=20)

$$APFD = 1 - \frac{1}{20} + \frac{1}{40} = 0.975$$

Evaluating Test Prioritization



1 fault detected after 2 test cases
2nd fault detected after 8 test cases (n=10)

$$APFD = 1 - \frac{2+8}{20} + \frac{1}{20} = 0.55$$

Test Case Prioritization

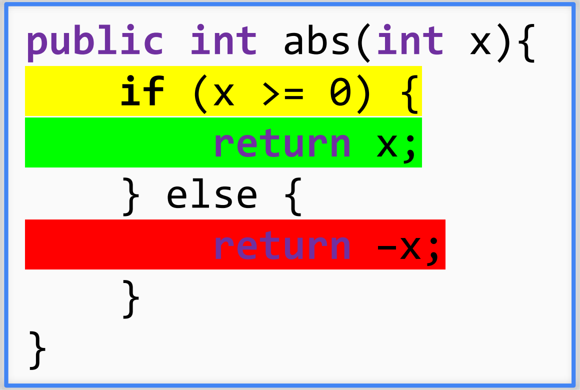
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	APFD
Version 1	☑	✗	☑	☑	☑	☑	☑	☑	☑	☑	-
Version 2	☑	✗	☑	☑	☑	☑	✗	☑	☑	☑	0.55
Version 3	☑	✗	☑	✗	☑	✗	☑	☑	☑	☑	0.45

Test Case Prioritization

	t_1	t_8	t_4	t_5	t_7	t_9	t_2	t_{10}	t_6	t_3	APFD
Version 1	☑	☑	☑	☑	☑	☑	✗	☑	☑	☑	-
Version 2	☑	☑	☑	☑	✗	☑	✗	☑	☑	☑	0.85
Version 3	☑	☑	✗	☑	☑	☑	✗	☑	✗	☑	0.8

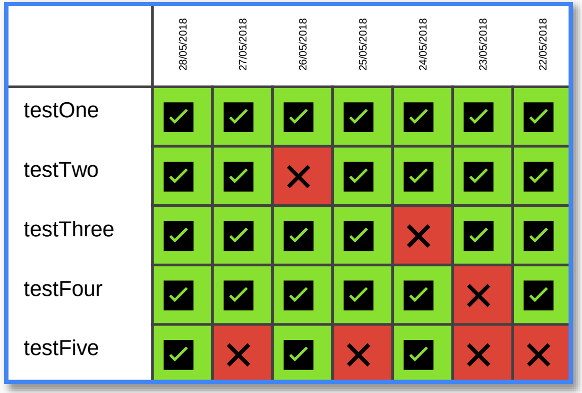
Techniques

Coverage-Based



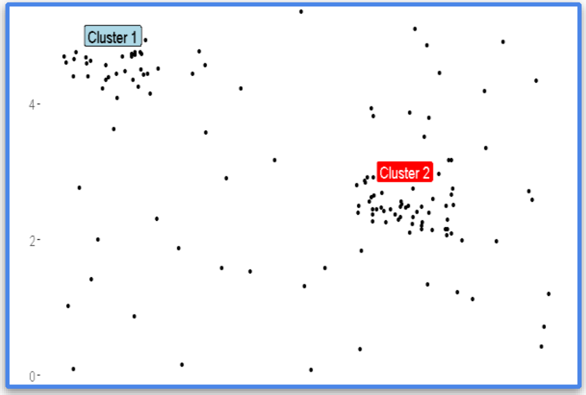
```
public int abs(int x){
    if (x >= 0) {
        return x;
    } else {
        return -x;
    }
}
```

History-Based



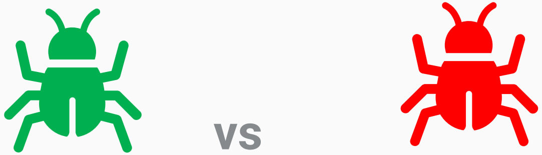
	28/05/2018	27/05/2018	26/05/2018	25/05/2018	24/05/2018	23/05/2018	22/05/2018
testOne	✓	✓	✓	✓	✓	✓	✓
testTwo	✓	✓	✗	✓	✓	✓	✓
testThree	✓	✓	✓	✓	✗	✓	✓
testFour	✓	✓	✓	✓	✓	✗	✓
testFive	✓	✗	✓	✗	✓	✗	✗

Cluster-Based



Evaluation

RQ1: How does the effectiveness of test case prioritization compare between a single real fault and a single mutant?



vs

RQ2: How does the effectiveness of test case prioritization compare between single faults and multiple faults?

vs

vs

Subjects

- **Defects4J**: Large repository containing 357 real faults from 5 open-source repositories [1]

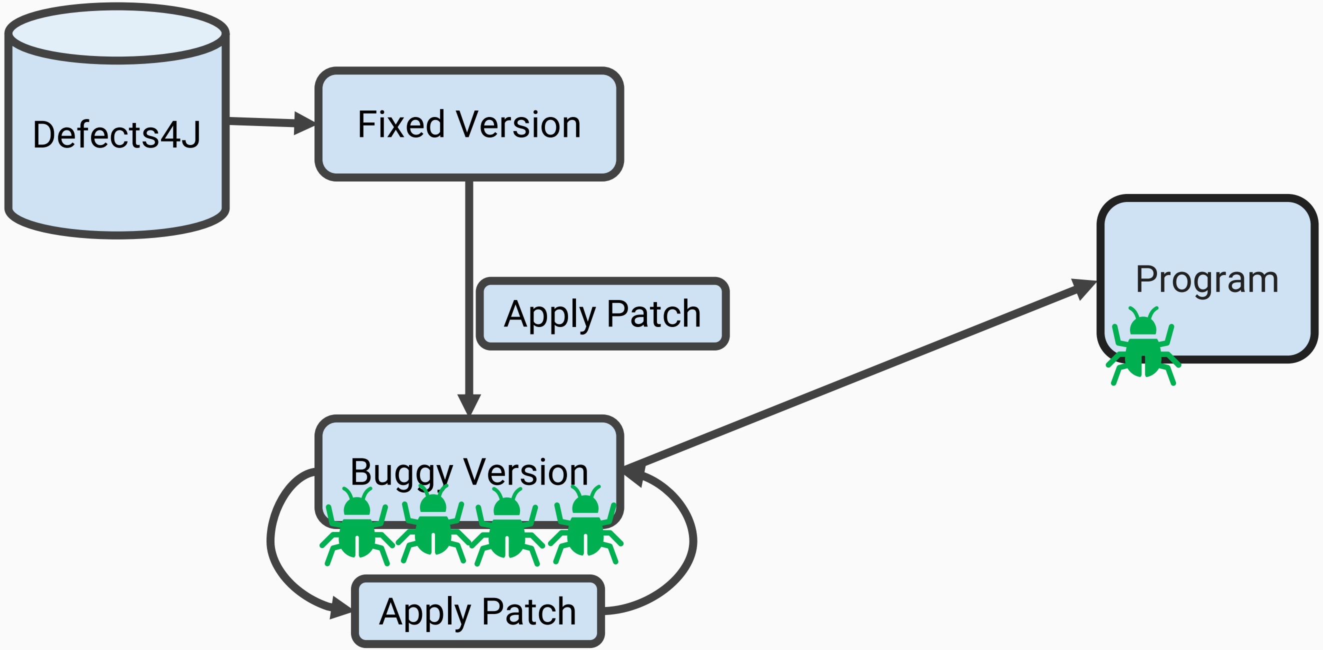
Project	GitHub	Number of Bugs	KLOC	Tests
JFreeChart	https://github.com/jfree/jfreechart	26	96	2,205
Closure Compiler	https://github.com/google/closure-compiler	133	90	7,927
Apache Commons Lang	https://github.com/apache/commons-lang	65	85	3,602
Apache Commons Math	https://github.com/apache/commons-math	106	28	4,130
Joda Time	https://github.com/JodaOrg/joda-time	27	22	2,245

- Contains developer written test suites

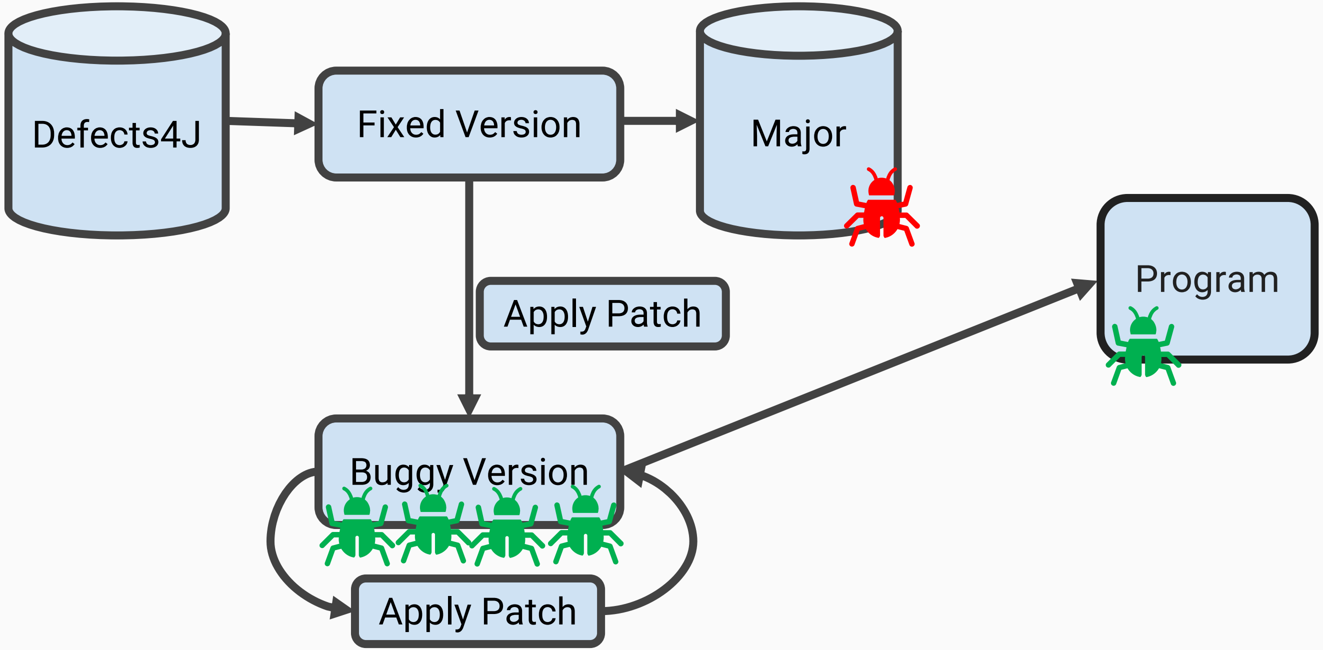
- Provides 2 versions of every subject – one buggy and one fixed

[1] https://github.com/rjust/defects4
[2] https://homes.cs.washington.edu/~mernst/pubs/bug-database-issta2014.pdfj

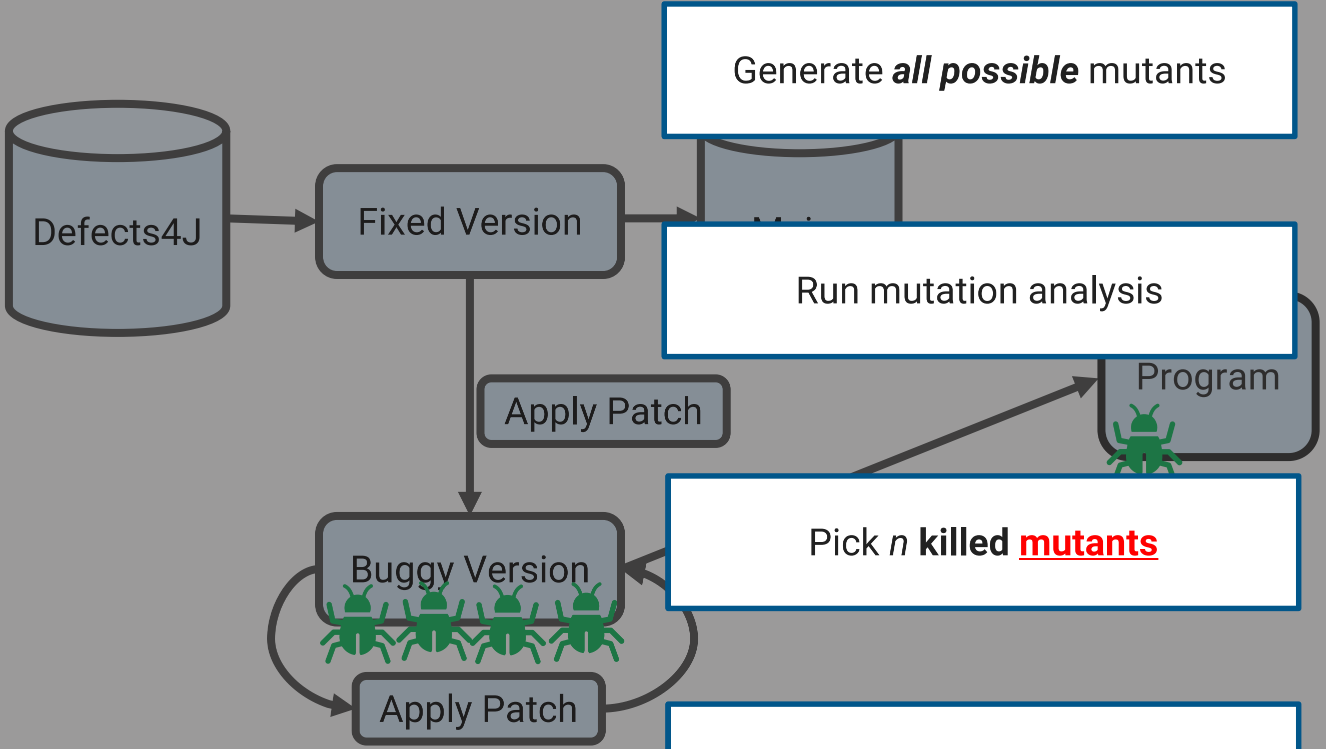
Experimental Process



Experimental Process



Experimental Process



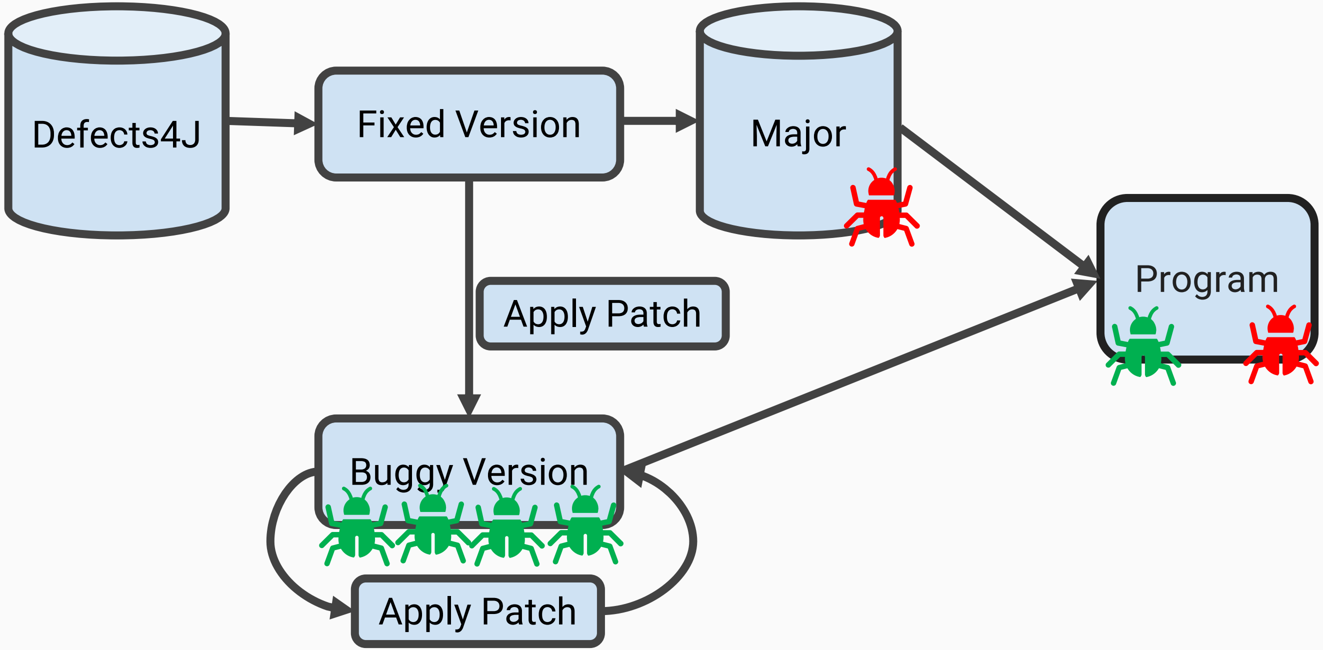
Generate ***all possible*** mutants

Run mutation analysis

Pick *n* **killed** **mutants**

Identify ***trigger tests***

Experimental Process



Defects4J

Fixed Version

Major

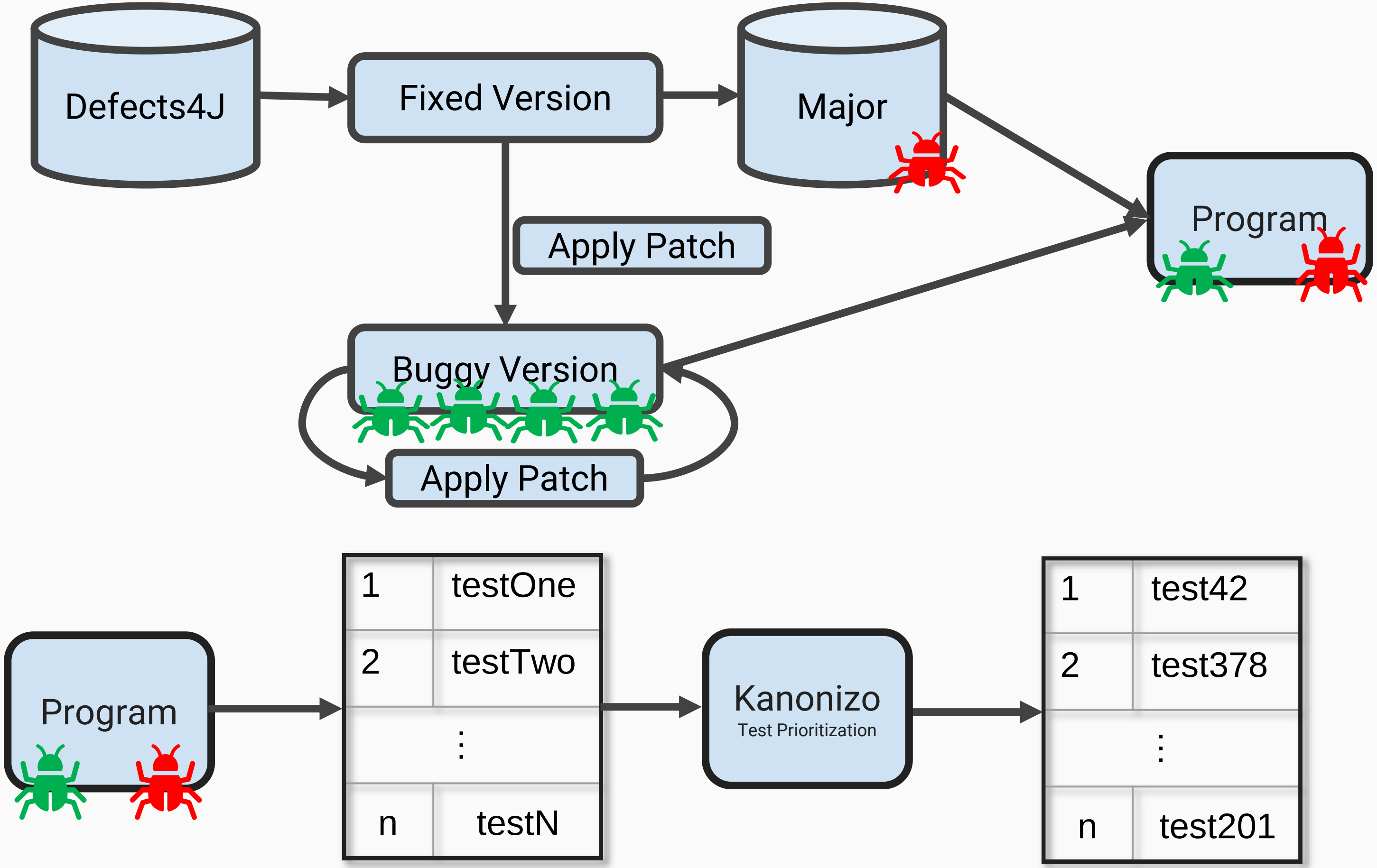
Apply Patch

Program

Buggy Version

Apply Patch

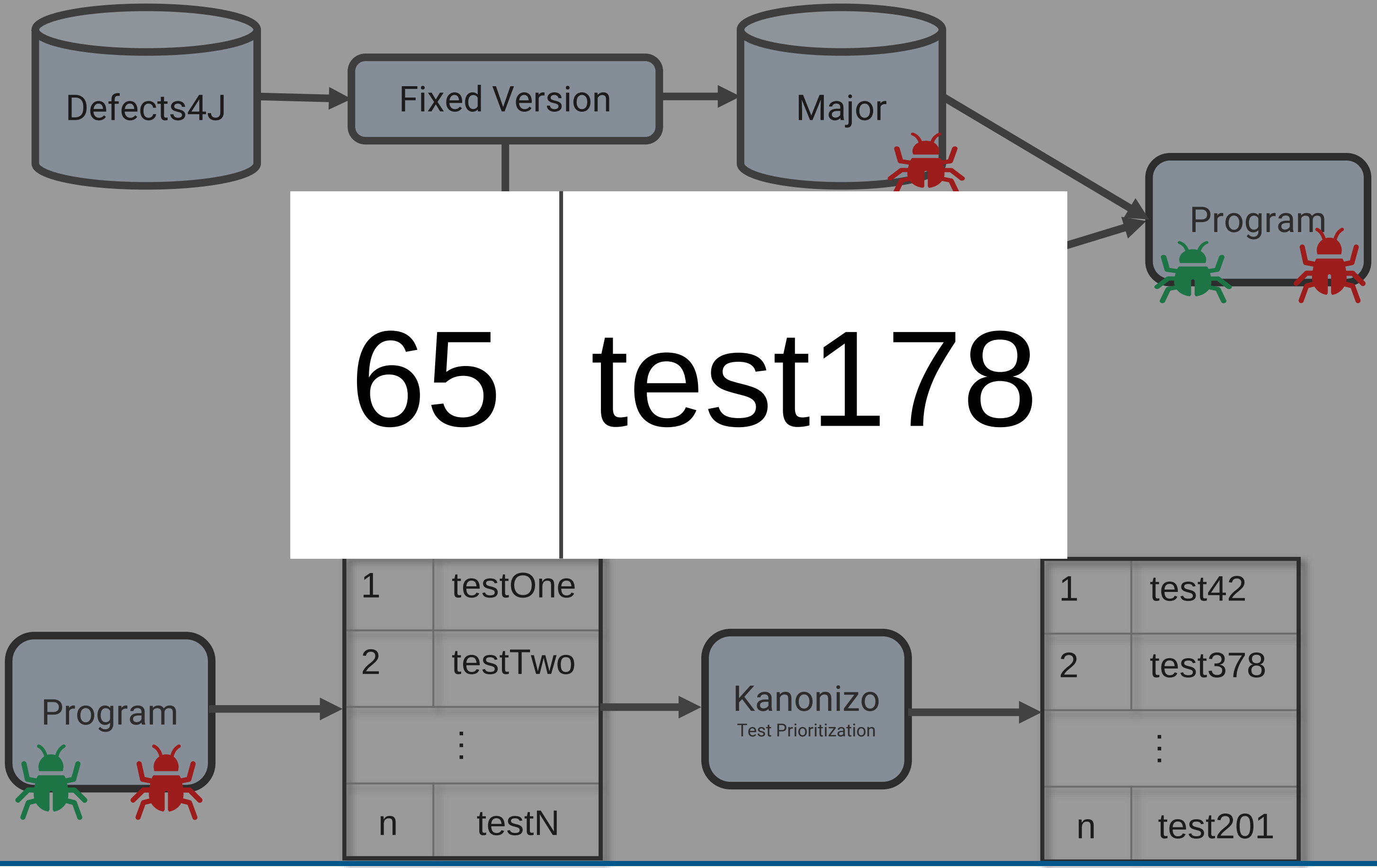
Experimental Process



1	testOne
2	testTwo
	⋮
n	testN

1	test42
2	test378
	⋮
n	test201

Experimental Process



65	test178

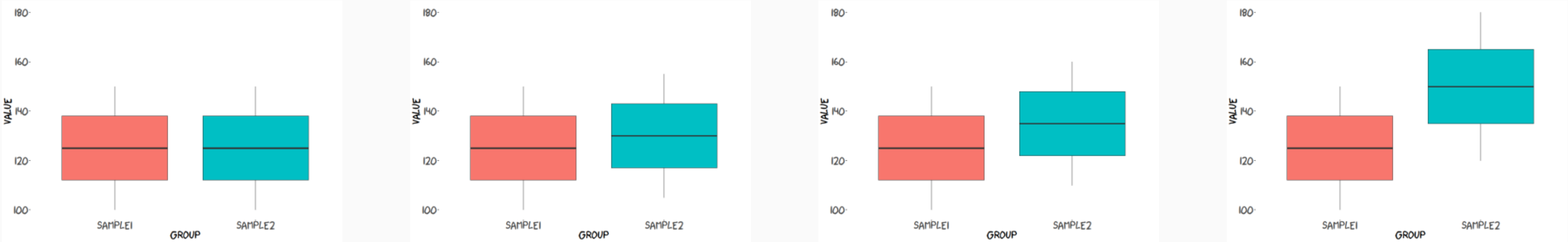
1	testOne
2	testTwo
	⋮
n	testN

1	test42
2	test378
	⋮
n	test201

Metrics

- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution p
  - Significant differences occur often when samples are large

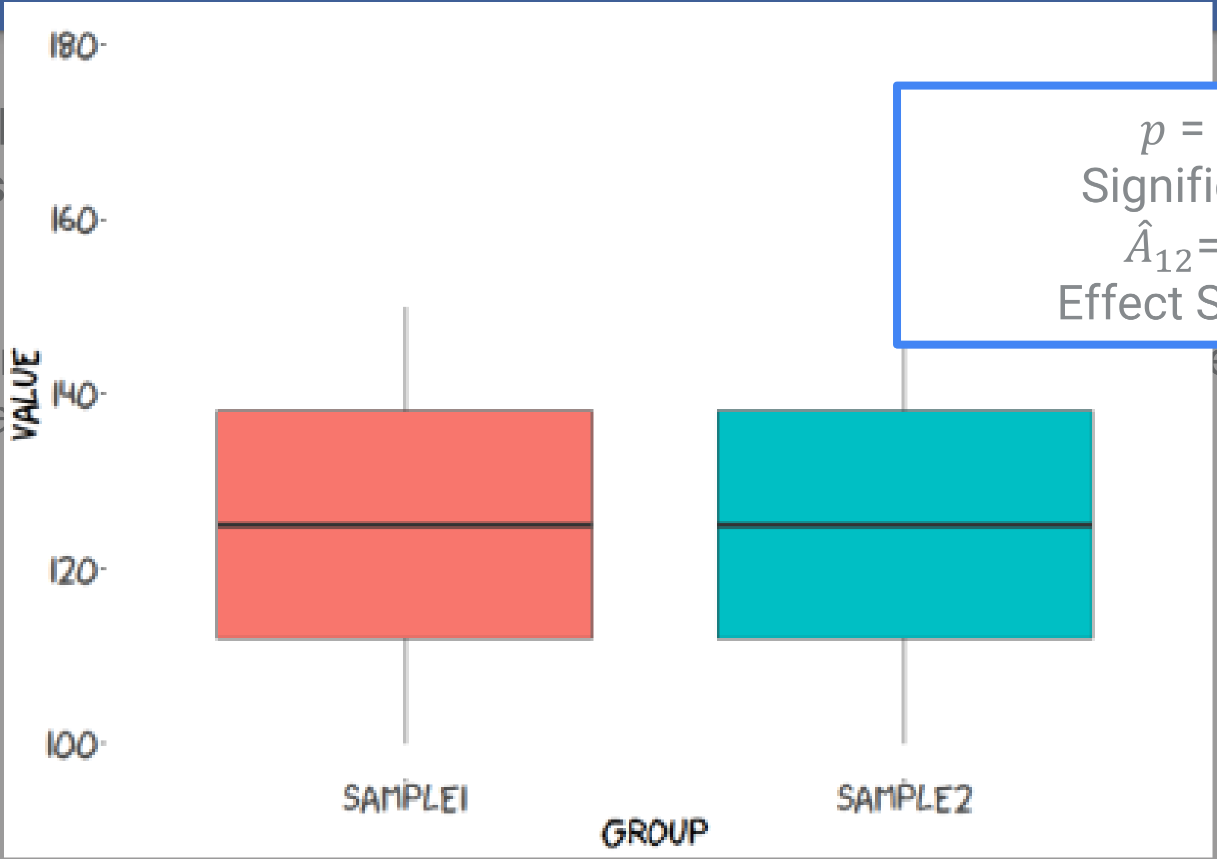
- **Vargha-Delaney** effect size calculates the *magnitude* of differences $\hat{A}_{12}$ – the practical difference between two samples



Metrics

- **Wi**
  dis

- **Va** erences – the
  pra

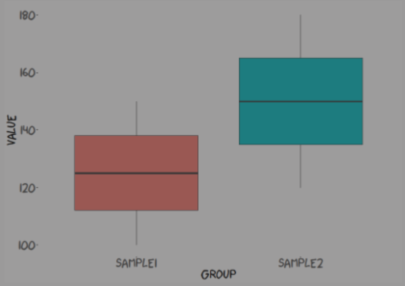


p = 0.5544

Significant = ✗

$\hat{A}_{12}$= 0.5007

Effect Size = None

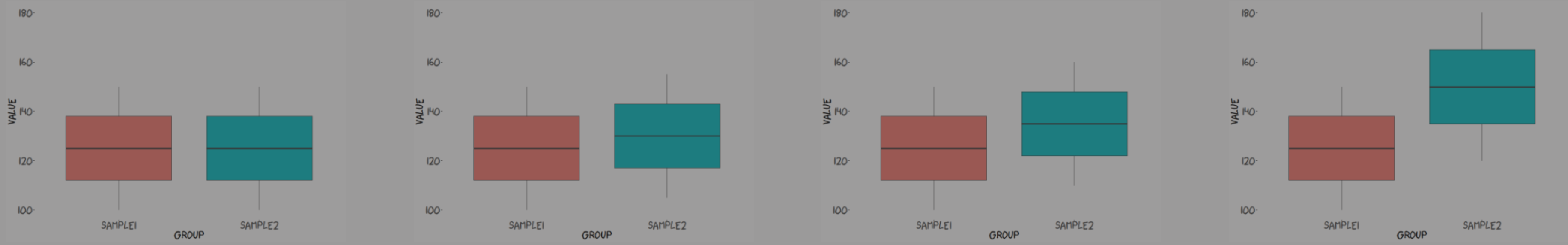


Metrics

- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution

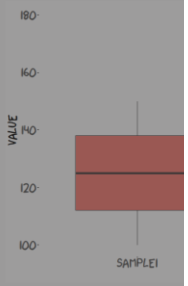
  - Significant differences occur often when samples are large

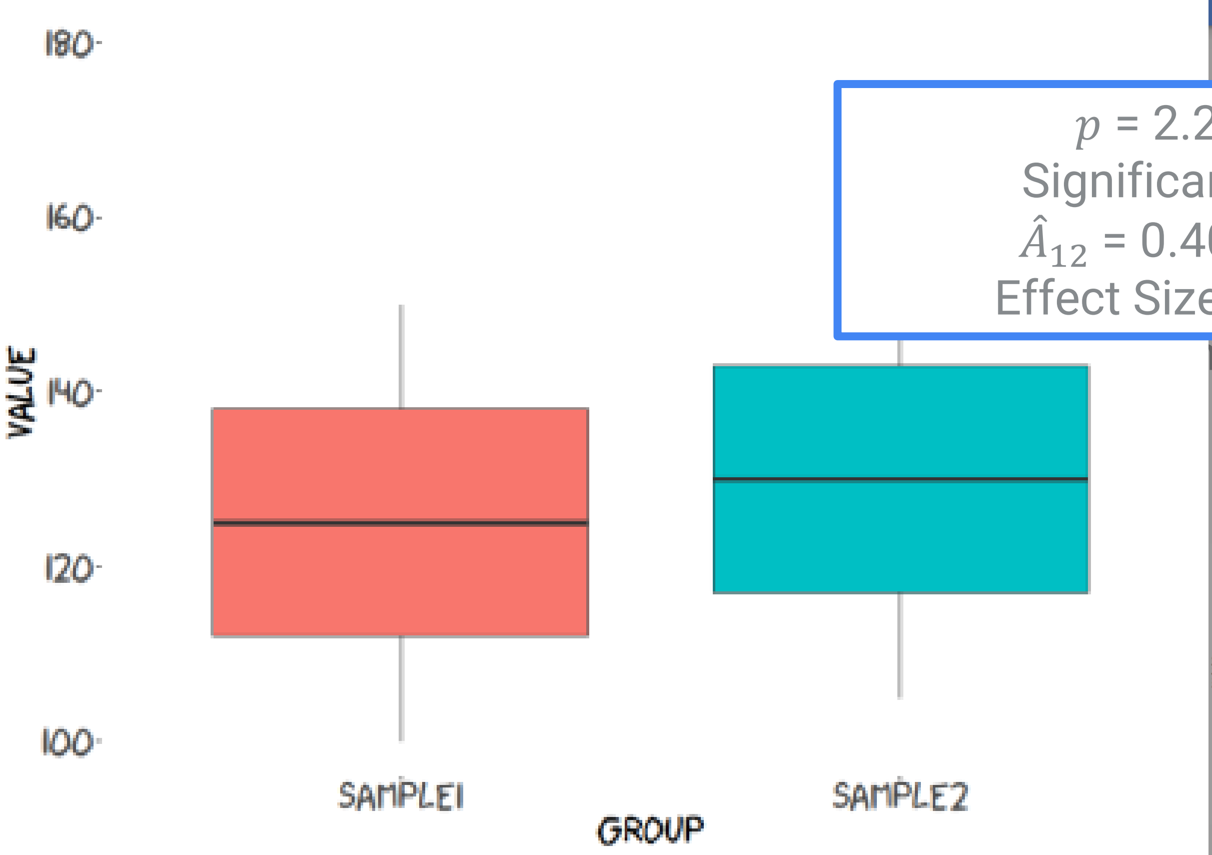
- **Vargha-Delaney** effect size calculates the *magnitude* of differences – the practical difference between two samples



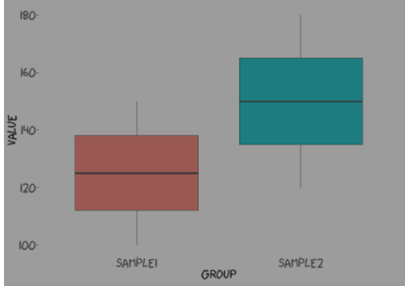
Metrics

- **Wilcox**
  distribu
  -
- **Vargha** nces – the
  practic





p = 2.2e-16
Significant = ✓
$\hat{A}_{12}$ = 0.4075059
Effect Size = Small

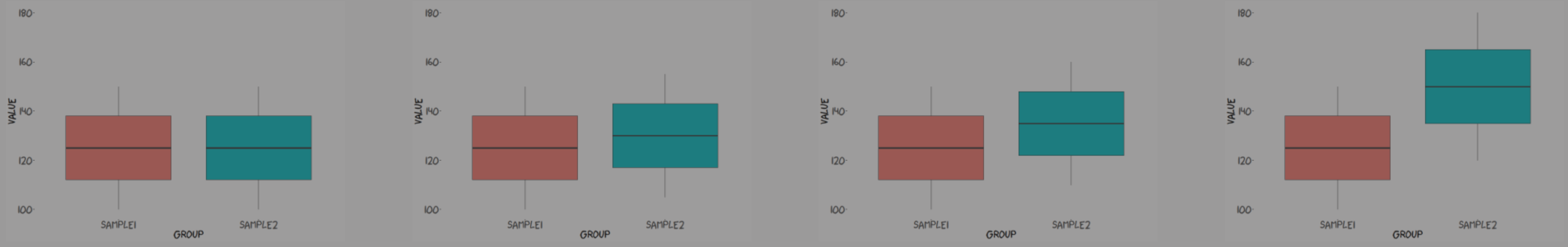


Metrics

- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution

    - Significant differences occur often when samples are large

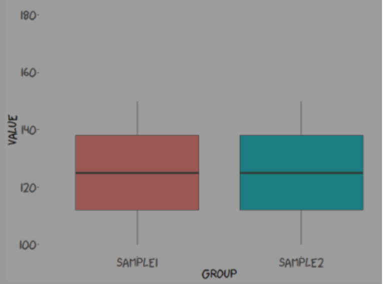
- **Vargha-Delaney** effect size calculates the *magnitude* of differences – the practical difference between two samples

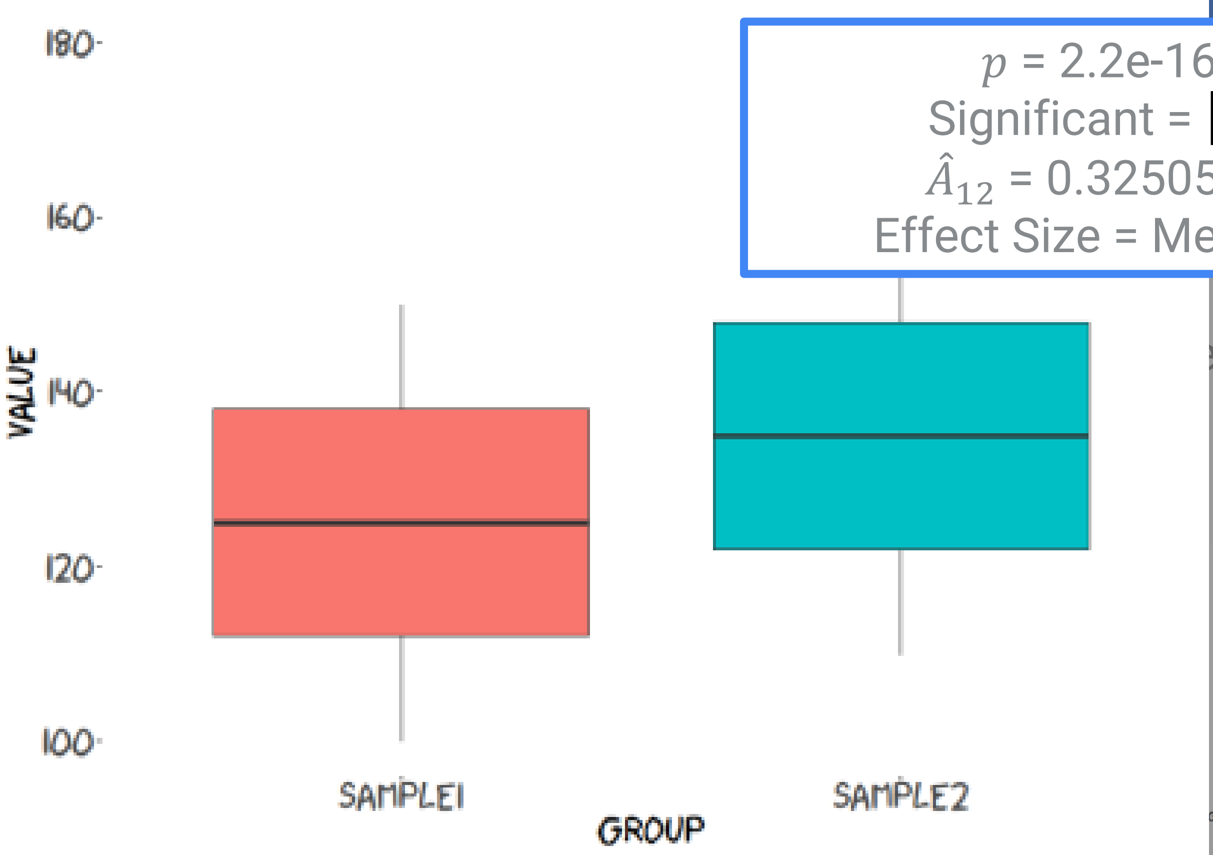


Metrics

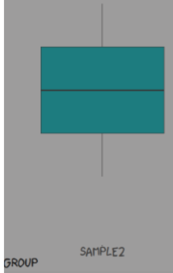
- **Wilcoxon U-Tes**
  distribution
  - Significa

- **Vargha-Delaney** e
  practical differe





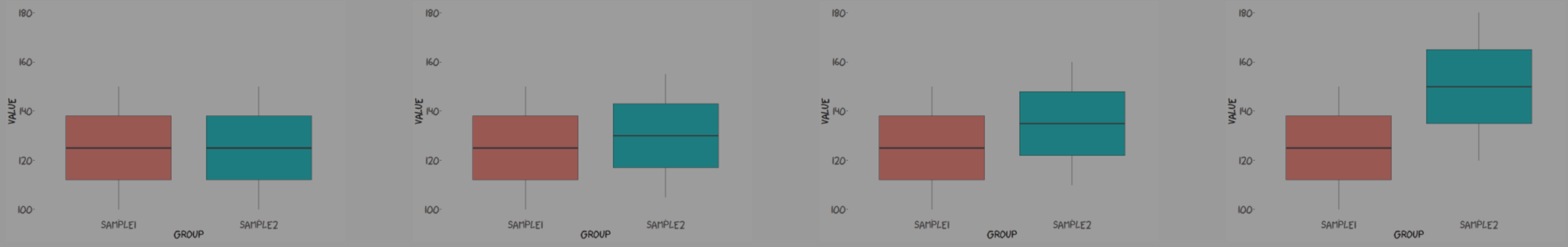
p = 2.2e-16
Significant = ✓
$\hat{A}_{12}$ = 0.3250598
Effect Size = Medium



Metrics

- **Wilcoxon U-Test** measures likelihood that 2 samples originate from the same distribution
  - Significant differences occur often when samples are large

- **Vargha-Delaney** effect size calculates the *magnitude* of differences – the practical difference between two samples



Metrics

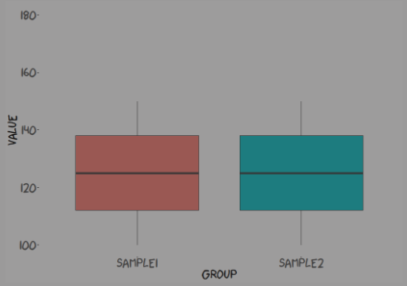
- **Wilc** distr
- **Vargha-Delaney** effect practical difference be

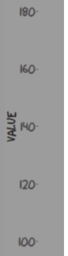
p = 2.2e-16

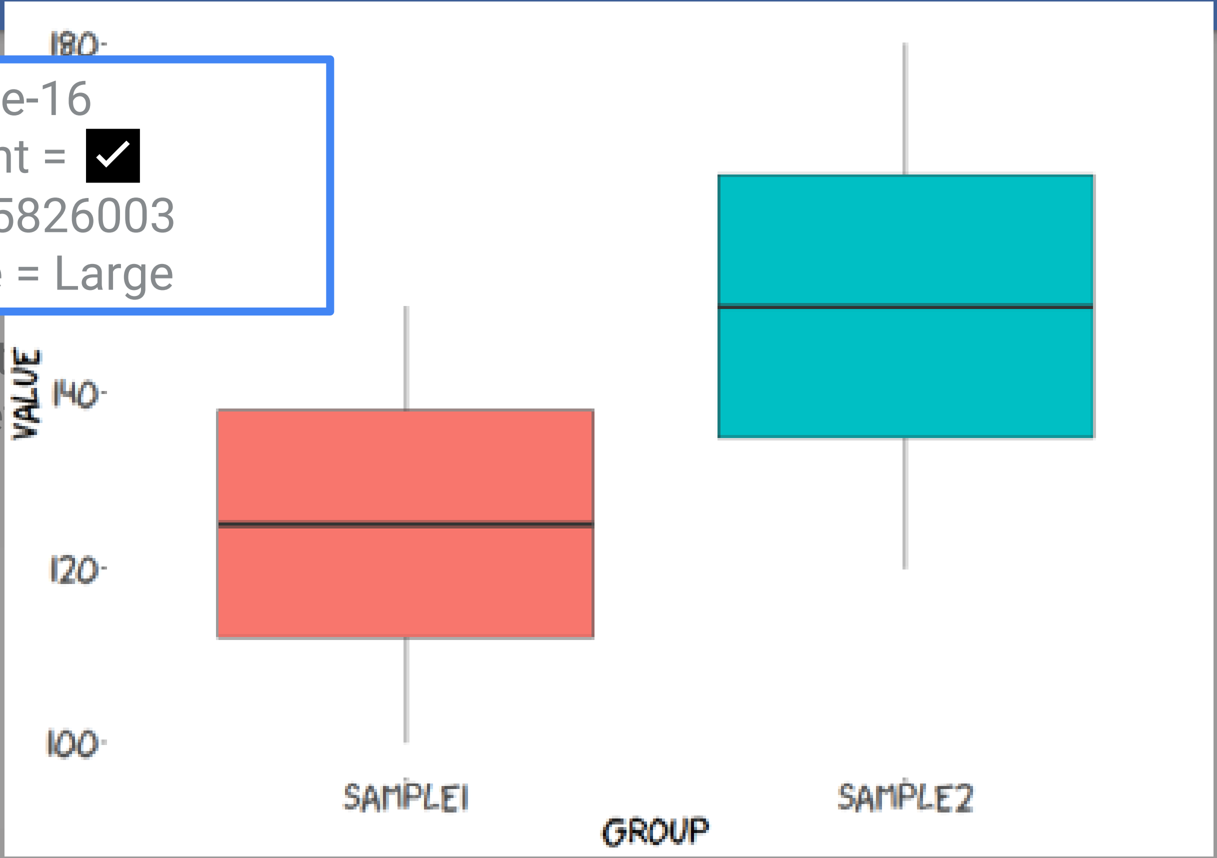
Significant = ☑

$\hat{A}_{12}$ = 0.005826003

Effect Size = Large







Comparisons

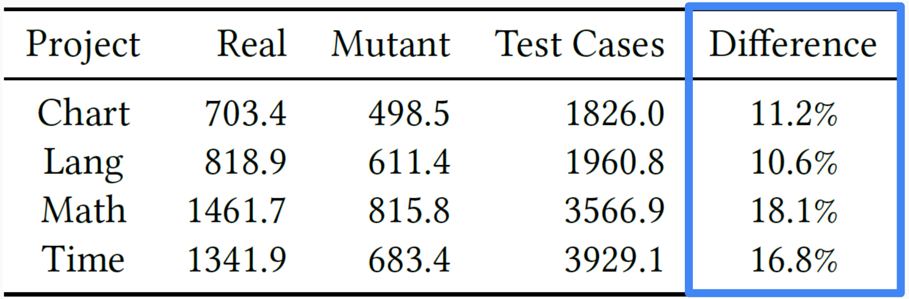
RQ1				RQ2				
Strategy 1	Strategy 2	Fault Type 1	Fault Type 2	Strategy 1	Strategy 2	Faults 1	Faults 2	Faults 3
A	A	**Real**	**Mutant**	A	A	1	5	10
A	B	**Real**	**Real**	A	B	1	5	10
A	B	**Mutant**	**Mutant**					

Results

RQ1: Real Faults vs Mutants

- APFD is **significantly higher** for **mutants** than **real faults** in all but one case

- On average, over **10% additional** test cases were required to find the **real faults**



Project	Real	Mutant	Test Cases	Difference
Chart	703.4	498.5	1826.0	11.2%
Lang	818.9	611.4	1960.8	10.6%
Math	1461.7	815.8	3566.9	18.1%
Time	1341.9	683.4	3929.1	16.8%

- For **real faults**, **3 out of 16** project/strategy combinations significantly improve over the baseline, compared to **10 out of 16** improvements for **mutants**

Results

RQ1: Real Faults vs Mutants

- APFD is **significantly higher** for **mutants** than **real faults** in all but one case

- On average, over **10% additional** test cases were required to find the **real faults**

Project	Real	Mutant	Test Cases	Difference
Chart	703.4	498.5	1826.0	11.2%
Lang	818.9	611.4	1960.8	10.6%
Math	1461.7	815.8	3566.9	18.1%
Time	1341.0	683.4	2020.1	16.8%

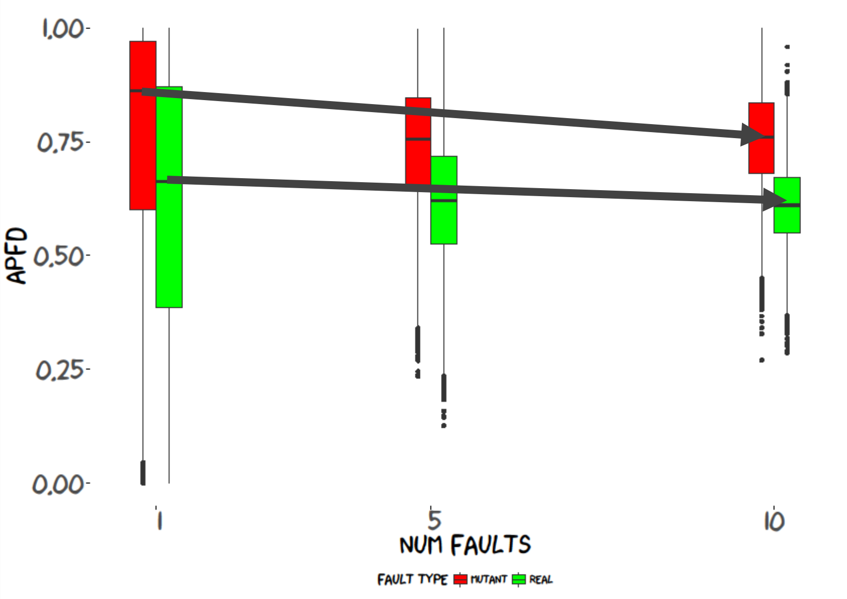
Test Case Prioritization is much more effective for mutants than real faults

- For **real faul** rove over the
  baseline, co

Results

RQ2: Single faults vs Multiple Faults

- Variance in APFD scores **<u>significantly</u>** reduces as more faults are introduced



- In **<u>37/40</u>** cases, median APFD *decreased* as more faults are introduced

  - APFD punishes test suites that are not able to find **<u>all</u>** faults

Results

RQ2: Single faults vs Multiple Faults

- However, **real faults** and **mutants** still disagree on the effectiveness of TCP techniques

- For **real faults**, there is very rarely any practical difference when including more faults
  - 17 of 40 comparisons are significant, of which 3 are **M**edium or **L**arge effect size

- For **mutants**, increasing the number of faults makes the results clearer
  - 35 of 40 comparisons are significant, of which 16 are **M**edium or **L**arge effect size
  - Effect size increases in **all but one** case for more faults

Results

RQ2: Single faults vs Multiple Faults

- However, **real faults** and **mutants** still disagree on the effectiveness of TCP techniques

- For **real faults**, there is very rarely any practical difference when including more faults

  - 17 of 40 comparisons are significant, of which 3 are Medium or Large effect size

- For **mutants**, increasing the number of faults makes the results clearer

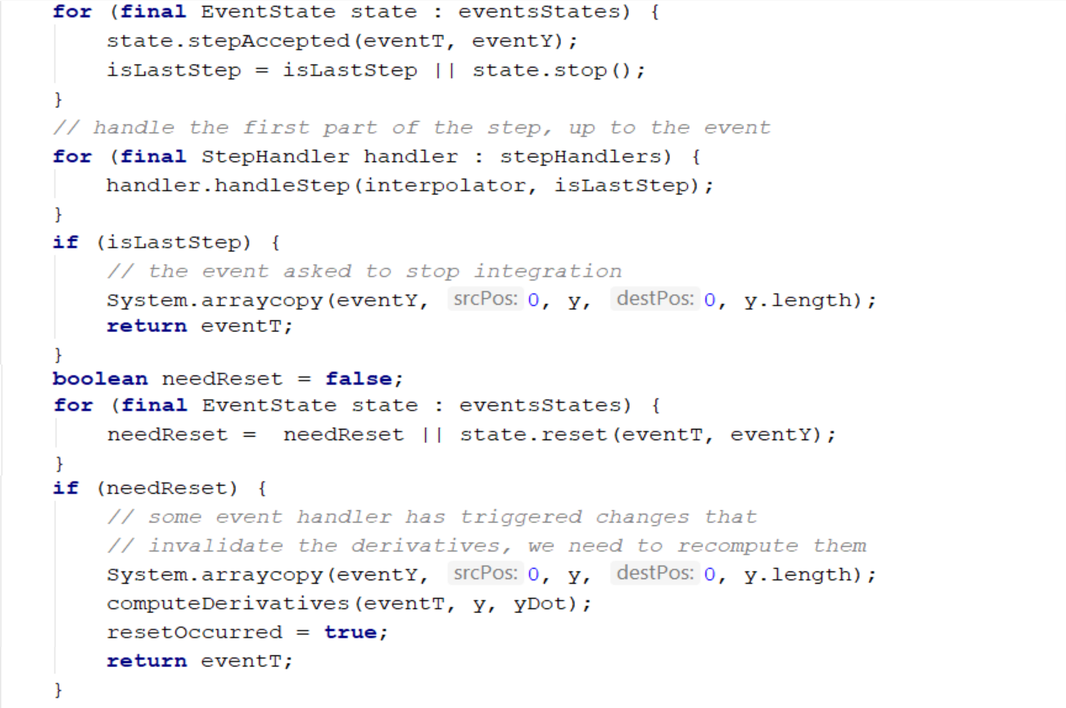
  - 35 of ge effect size

  - Effect

Using more faults lessens the effect of randomness, but still does not make mutants and real faults consistent

Real Faults vs Mutants

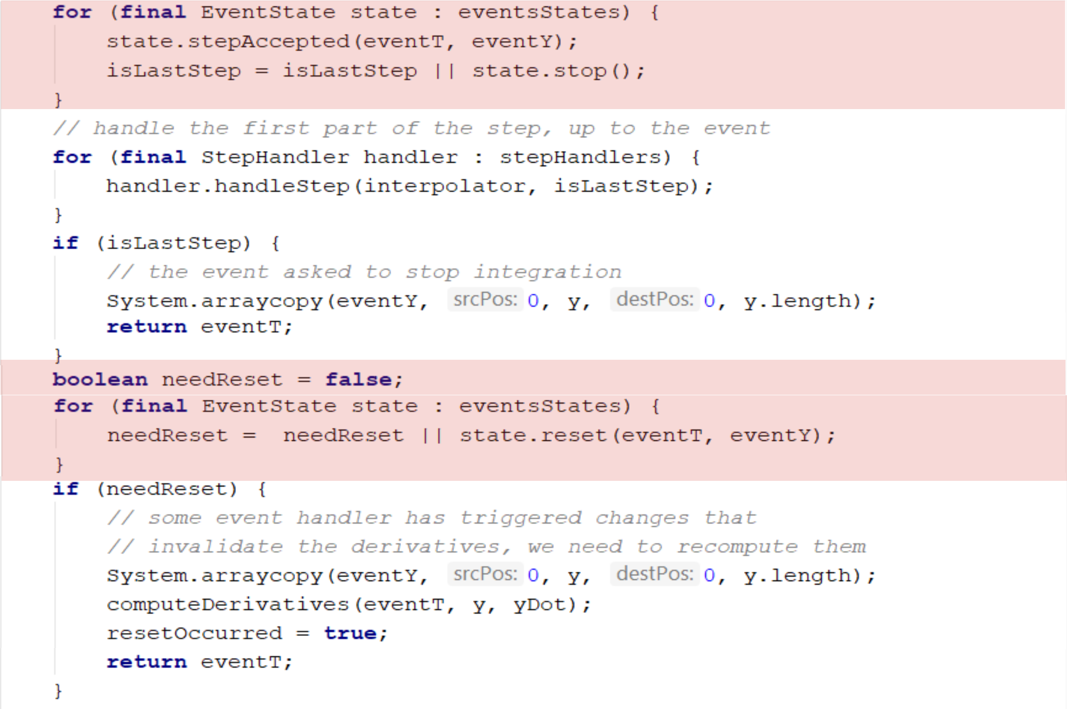
- Real faults are much more complex than mutants



```java
for (final EventState state : eventsStates) {
    state.stepAccepted(eventT, eventY);
    isLastStep = isLastStep || state.stop();
}
// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    handler.handleStep(interpolator, isLastStep);
}
if (isLastStep) {
    // the event asked to stop integration
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    return eventT;
}
boolean needReset = false;
for (final EventState state : eventsStates) {
    needReset =  needReset || state.reset(eventT, eventY);
}
if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
    return eventT;
}
```

Real Faults vs Mutants

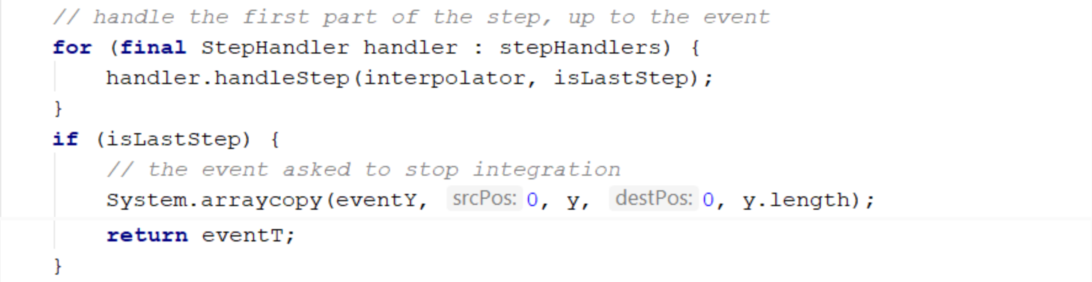
- Real faults are much more complex than mutants



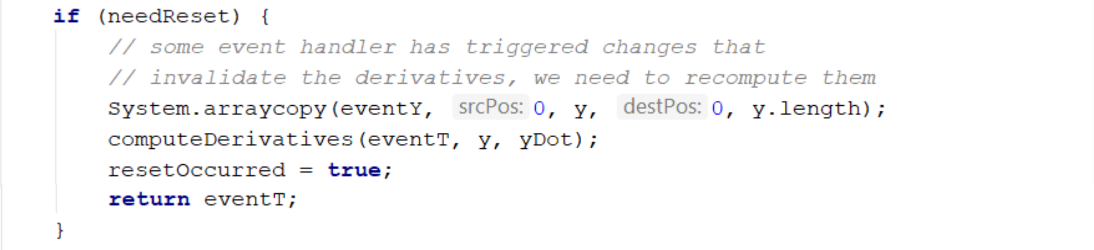
```
for (final EventState state : eventsStates) {
    state.stepAccepted(eventT, eventY);
    isLastStep = isLastStep || state.stop();
}
// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    handler.handleStep(interpolator, isLastStep);
}
if (isLastStep) {
    // the event asked to stop integration
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    return eventT;
}
boolean needReset = false;
for (final EventState state : eventsStates) {
    needReset =  needReset || state.reset(eventT, eventY);
}
if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
    return eventT;
}
```

Real Faults vs Mutants

- Real faults are much more complex than mutants



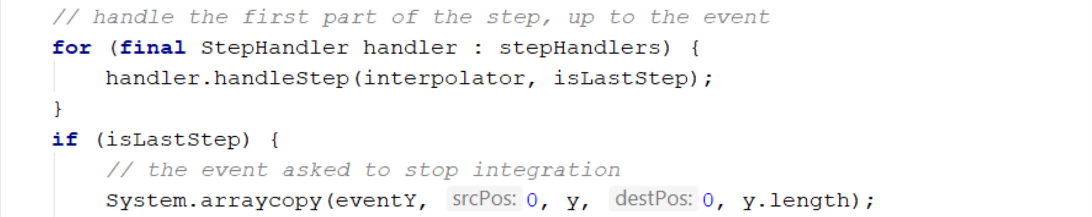
```
// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    handler.handleStep(interpolator, isLastStep);
}
if (isLastStep) {
    // the event asked to stop integration
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    return eventT;
}
```



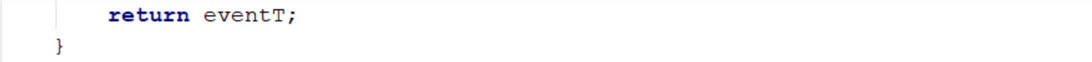
```
if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
    return eventT;
}
```

Real Faults vs Mutants

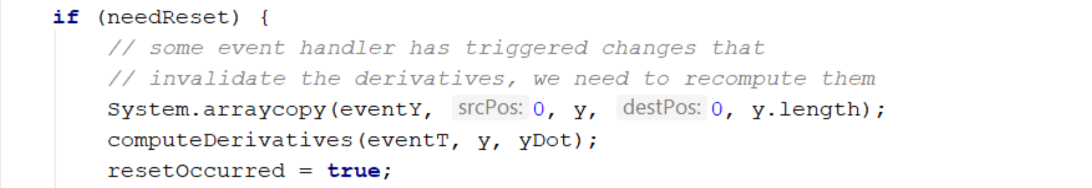
- Real faults are much more complex than mutants



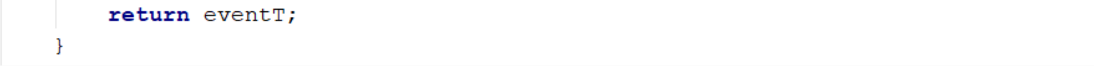
```
// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    handler.handleStep(interpolator, isLastStep);
}
if (isLastStep) {
    // the event asked to stop integration
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
```



```
    return eventT;
}
```



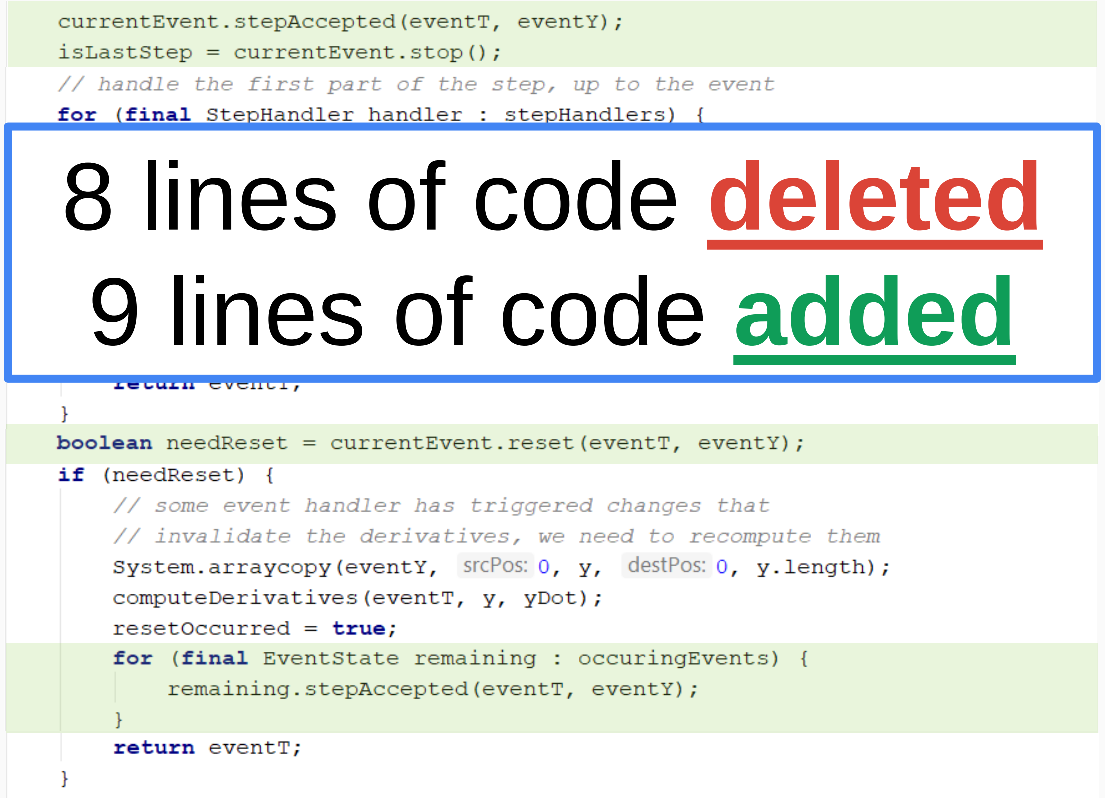
```
if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
```



```
    return eventT;
}
```

Real Faults vs Mutants

- Real faults are much more complex than mutants



```
currentEvent.stepAccepted(eventT, eventY);
isLastStep = currentEvent.stop();
// handle the first part of the step, up to the event
for (final StepHandler handler : stepHandlers) {
    return eventT;
}
boolean needReset = currentEvent.reset(eventT, eventY);
if (needReset) {
    // some event handler has triggered changes that
    // invalidate the derivatives, we need to recompute them
    System.arraycopy(eventY, srcPos: 0, y, destPos: 0, y.length);
    computeDerivatives(eventT, y, yDot);
    resetOccurred = true;
    for (final EventState remaining : occuringEvents) {
        remaining.stepAccepted(eventT, eventY);
    }
    return eventT;
}
```

8 lines of code <u>deleted</u>

9 lines of code <u>added</u>

Real Faults vs Mutants

- Real faults are much more complex than mutants
  - On average, fixing a **real fault** added 1.98 lines and removed 7.2
  - Fixing a **mutant** is always **max** +/- 1 line

```
boolean needsReset = true;
```

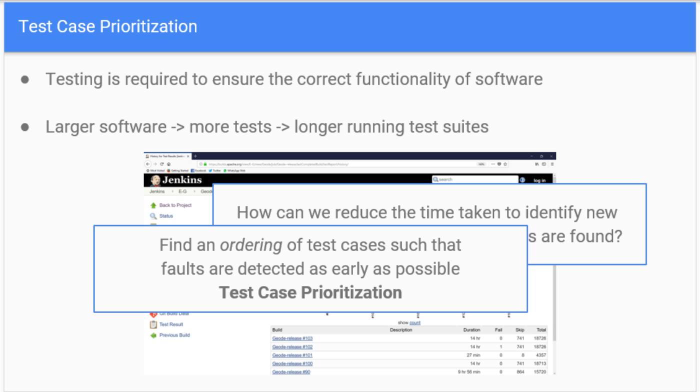
Real Faults vs Mutants

- Real faults are much more complex than mutants
  - On average, fixing a **real fault** added 1.98 lines and removed 7.2
  - Fixing a **mutant** is always **max** +/- 1 line

```
boolean needsReset = false;
```

- This results in **more** test cases detecting **mutants**
  - On average, 3.18 test cases detected single **real faults**
  - Meanwhile, 57.38 test cases detected single **mutants**

Summary



Test Case Prioritization

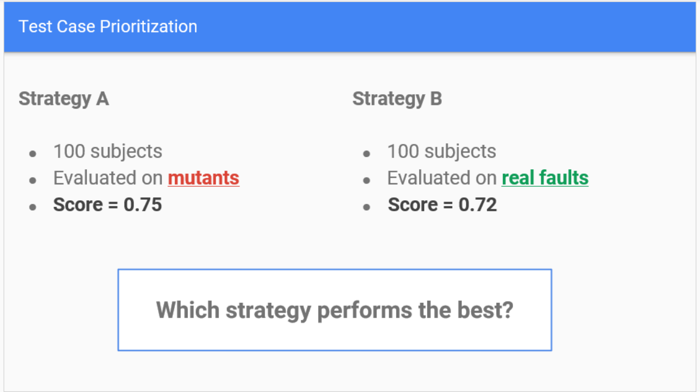
- Testing is required to ensure the correct functionality of software
- Larger software -> more tests -> longer running test suites

How can we reduce the time taken to identify new

s are found?

Find an *ordering* of test cases such that faults are detected as early as possible

Test Case Prioritization



Test Case Prioritization

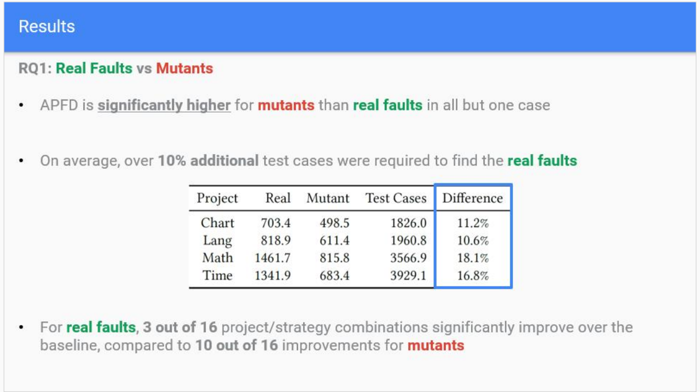
Strategy A

- 100 subjects
- Evaluated on **mutants**
- **Score = 0.75**

Strategy B

- 100 subjects
- Evaluated on **real faults**
- **Score = 0.72**

Which strategy performs the best?



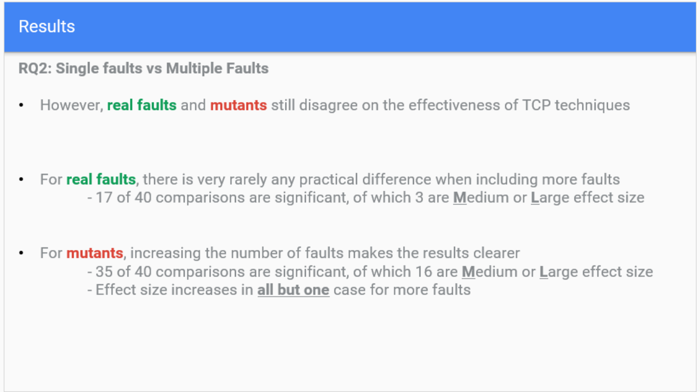
Results

RQ1: Real Faults vs Mutants

- APFD is **significantly higher** for **mutants** than **real faults** in all but one case
- On average, over **10% additional** test cases were required to find the **real faults**

Project	Real	Mutant	Test Cases	Difference
Chart	703.4	498.5	1826.0	11.2%
Lang	818.9	611.4	1960.8	10.6%
Math	1461.7	815.8	3566.9	18.1%
Time	1341.9	683.4	3929.1	16.8%

- For **real faults**, **3 out of 16** project/strategy combinations significantly improve over the baseline, compared to **10 out of 16** improvements for **mutants**



Results

RQ2: Single faults vs Multiple Faults

- However, **real faults** and **mutants** still disagree on the effectiveness of TCP techniques
- For **real faults**, there is very rarely any practical difference when including more faults
  - 17 of 40 comparisons are significant, of which 3 are Medium or Large effect size
- For **mutants**, increasing the number of faults makes the results clearer
  - 35 of 40 comparisons are significant, of which 16 are Medium or Large effect size
  - Effect size increases in **all but one** case for more faults

Tool:
https://github.com/kanonizo/kanonizo

Data:
https://bitbucket.org/djpaterson/ast2018_data