



Testing the tests with Mutalk in 2024

guillermo.polito@inria.fr

Guillermo Polito - ESUG'24



Quick About Me

guillermo.polito@inria.fr
@guillep



- **Now:** Researcher at Inria - Lille
- Pharo Contributor since ~2010

- **Keywords:** compilers, testing, test generation
- **Interests:** tooling, benchmarking, 日本語, board games, batman, concurrency

If any of that interests you, come talk to me!



Inria

Automated Tests

SetTest >> testAdd

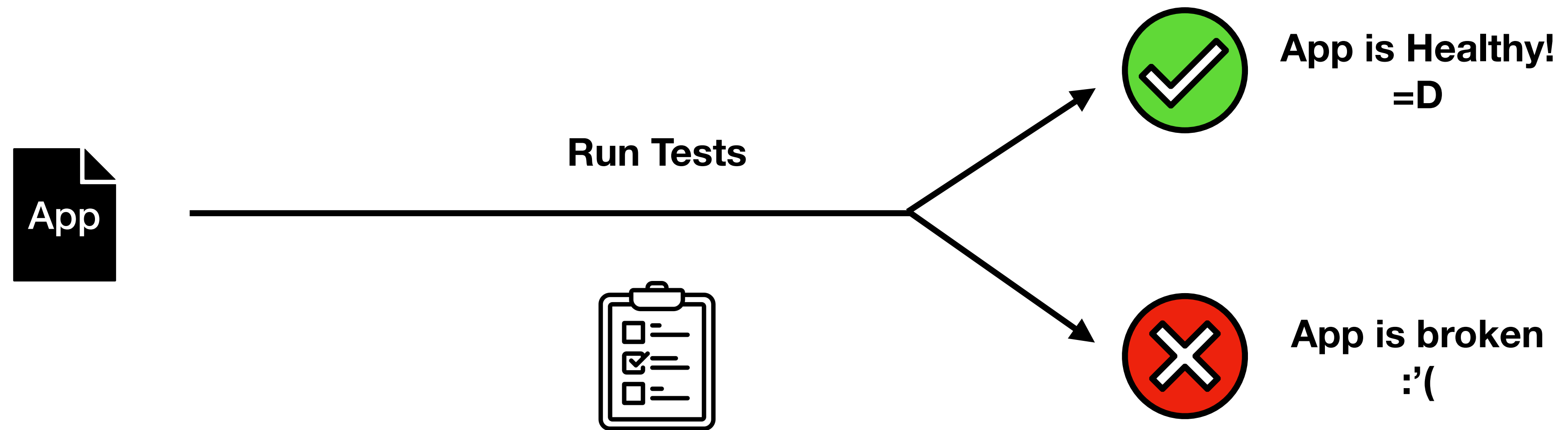
```
| aSet |  
"Context"  
aSet := Set new.
```

```
"Stimuli"  
aSet add: 5.  
aSet add: 5.
```

```
"Check"  
self assert: aSet size equals: 1.
```

in this context
when this happens
then this should happen

We love tests



What is a good test?

“A good test is a test that catches bugs”

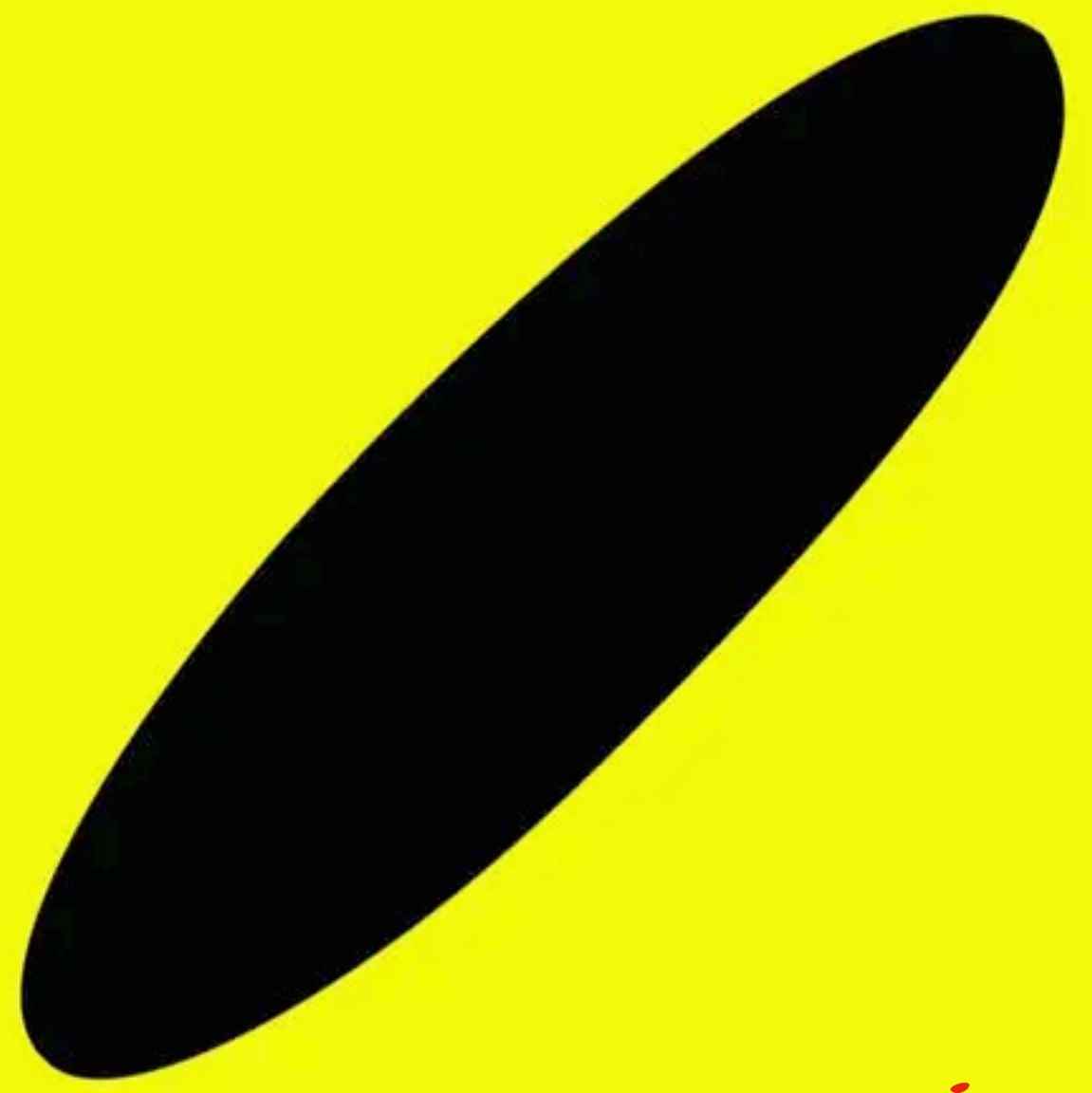
- me

WHO

WATCHES

THE

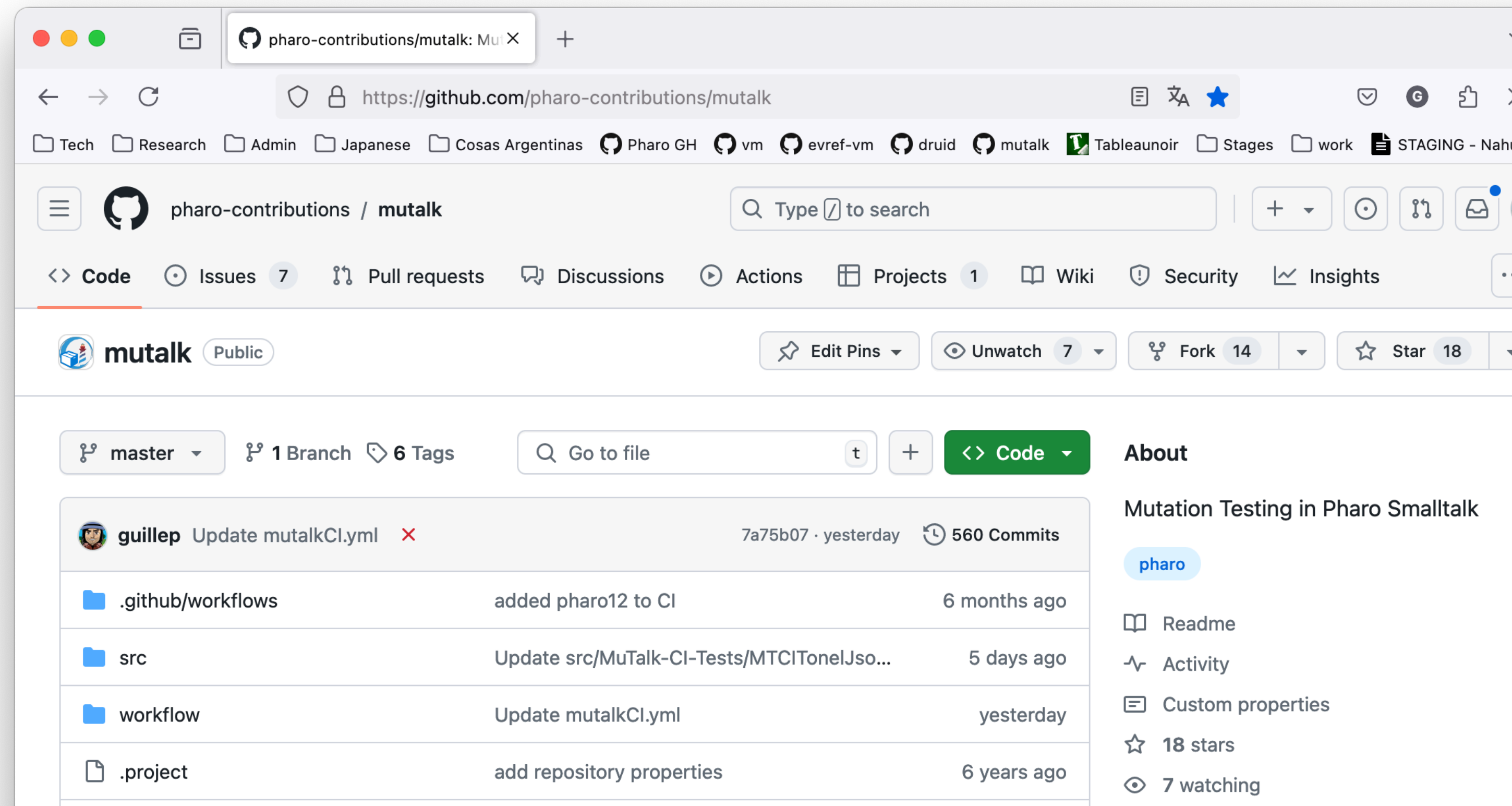
TESTS





Mutalk: Mutation testing for Pharo

- Originally developed in Pharo 1.1 in Argentina (Chillo, Brunstein, Wilkinson)
- Presented at ESUG'09
- Pharo 9 to 12 !



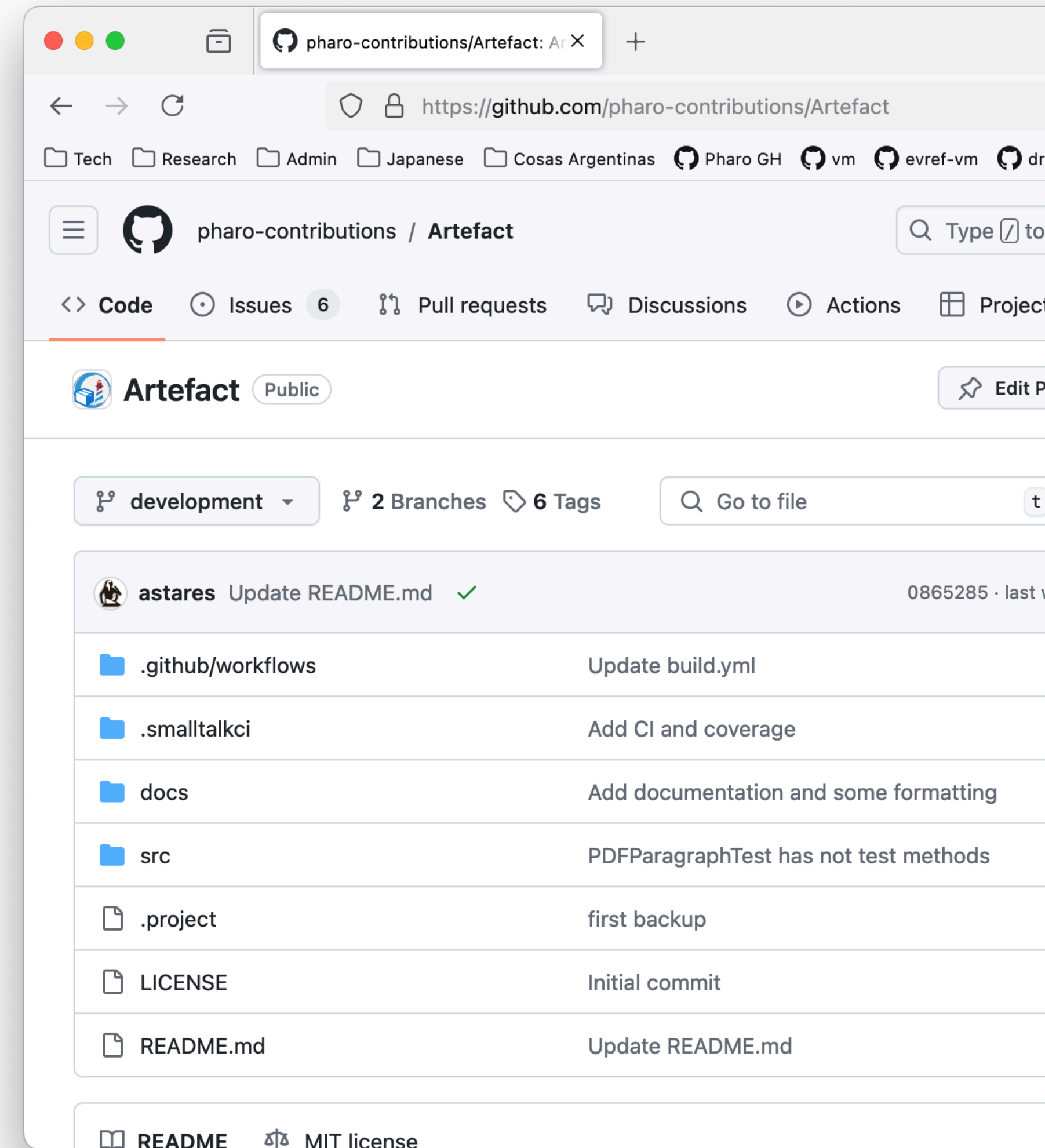
Coverage vs Mutations



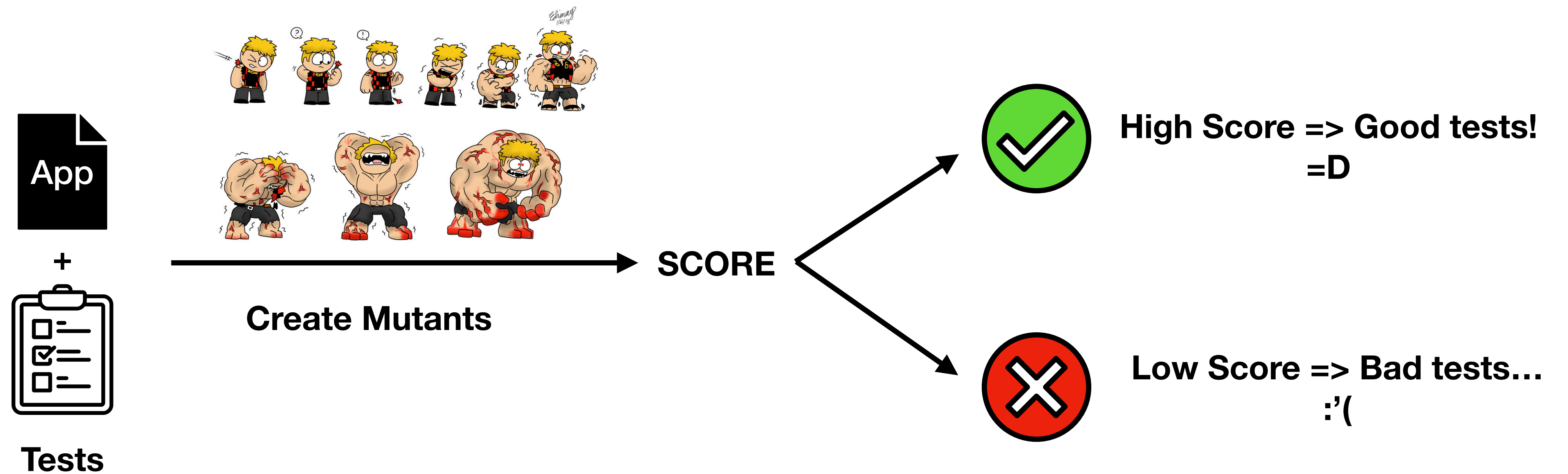
30%
Code
Coverage



4%
Mutation
Coverage



Mutation testing in a nutshell







These kind of mutant

- Introduce artificial bugs
- See if the test suite detects them
- What kind of bugs?



Competent developer hypothesis

- Developers are **capable people**
- Mistakes/bugs are **small details** easily overseen. E.g.,
 - Missing +/- 1 in a loop
 - An inverted conditional
 - Signed/unsigned

Thousands of simple mutations

- Control flow bugs
- Arithmetic bugs
- Logic bugs
- Overflow bugs
- Typos

Inspector on 3090 mutants, 1599 killed, 1491 alive...

a MTGeneralResult (3090 mut...)

Surviving Mutants | Killed Mutants | Terminated Mutants | Excluded Tests | Raw | Breakpoints | Meta

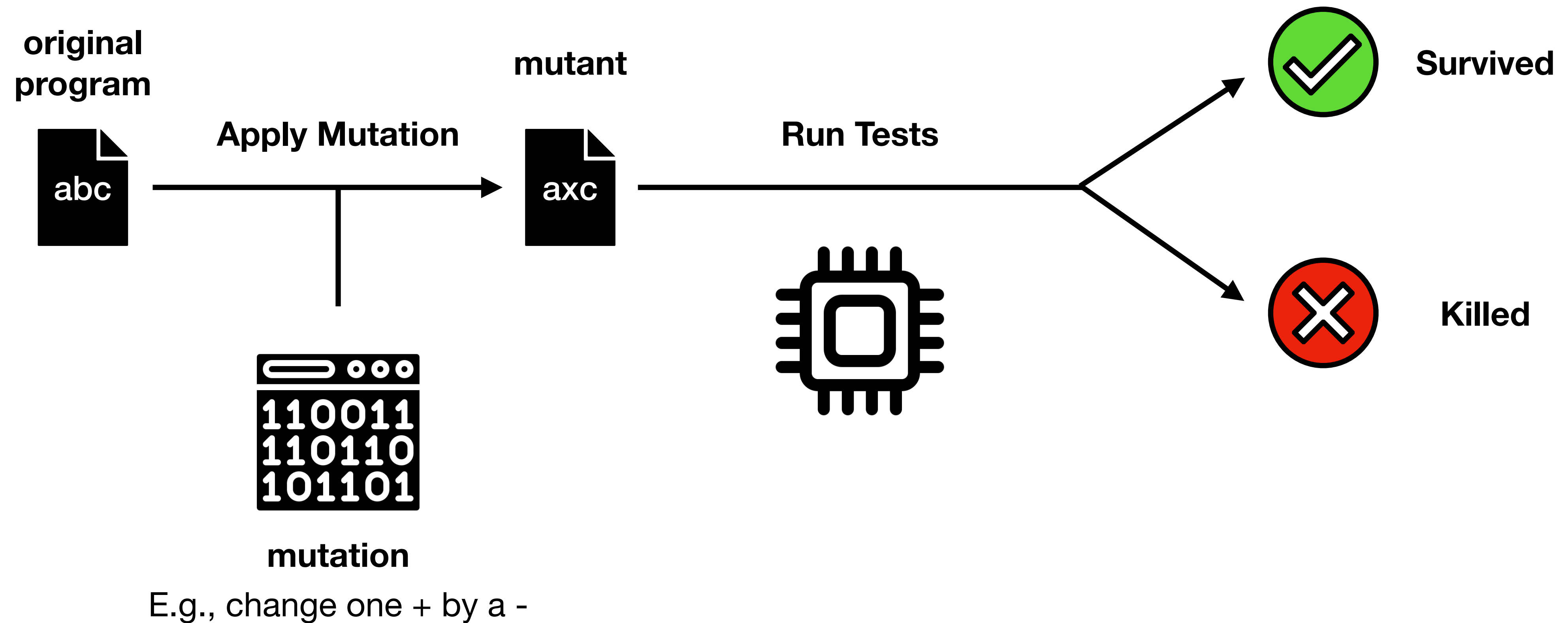
Results

- 1 Replace #ifTrue: receiver with false in MicAnchorBlock>>#printOn:
- 2 Replace detect: block with [:each | true] when #detect:ifNone: in Class>>#documentExampleCode
- 3 Replace #ifFalse: receiver with false in MicInlineDelimiter class>>#initializeDelimiters
- 4 Replace #ifTrue:ifFalse: receiver with false in MicMicrodownTextualBuilder>>#mathblock:firstLineAssociations:withCaption:
- 5 Decrease a literal integer in Package>>#buildMicroDownUsing:withComment:
- 6 Replace #ifTrue: with #ifFalse: in BaselineOf class>>#buildMicroDownUsing:withComment:
- 7 Replace #= with #'~=' in MicInlineDelimiter class>>#initializeDelimiters
- 8 Replace #or: with #and: in MicFileResourceReference>>#loadChildren

```
mathblock: aString firstLineAssociations: aCol withCaption: aCaptionBlock value ] ]
self raw: MathOpeningBlockMarkup.
aCol
do: [ :each |
self
raw: each key;
raw: '='.
each key = #caption
ifTrue: [ aCaptionBlock value ]
ifFalse: [ "so that we can put format such as bold"
self raw: each value value ] ]
separatedBy: [ self raw: '&' ].
self newline
```

```
mathblock: aString firstLineAssociations: aCol withCaption: aCaptionBlock value ] ]
self raw: MathOpeningBlockMarkup.
aCol
do: [ :each |
self
raw: each key;
raw: '='.
false
ifTrue: [ aCaptionBlock value ]
ifFalse: [ "so that we can put format such as bold"
self raw: each value value ] ]
separatedBy: [ self raw: '&' ].
self newline
```

Mutation Analysis



The insight

- Survived mutants were either

1. not covered

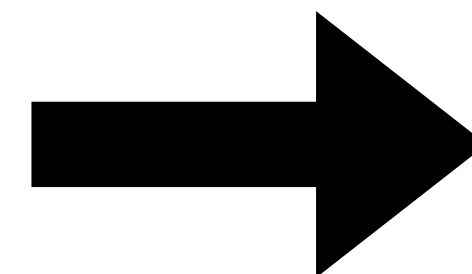
2. not asserted

3. or semantically **equivalent**.

E.g. $A+B=A-B$ if $B=0$ always



improve your tests!!



bias our results

Mutation Score

- Run each mutation independently
- Score:

$$\frac{\#Killed}{\#Mutants}$$

THE Problem of Mutation Analysis

$$\text{Runtime} = \text{Time}(\text{tests}) * \#\text{Mutants}$$

Optimizing of Mutation Analysis

$$\text{Runtime} = \text{Time}(\text{tests}) * \#\text{Mutants}$$

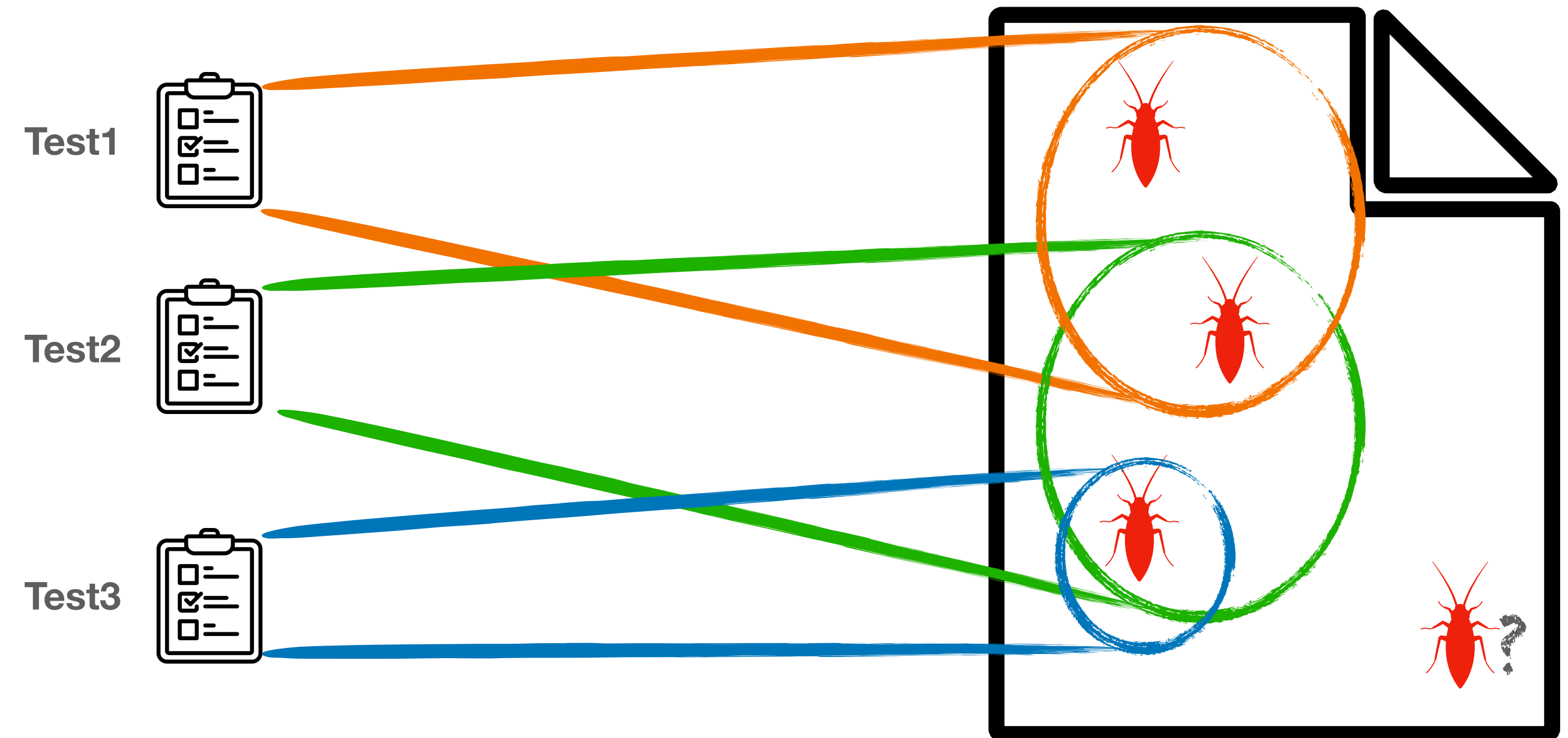
Can we run **less tests**?

Can we run **less mutants**?

Ideally... no impact on score!

Selecting Mutants and Tests from Coverage

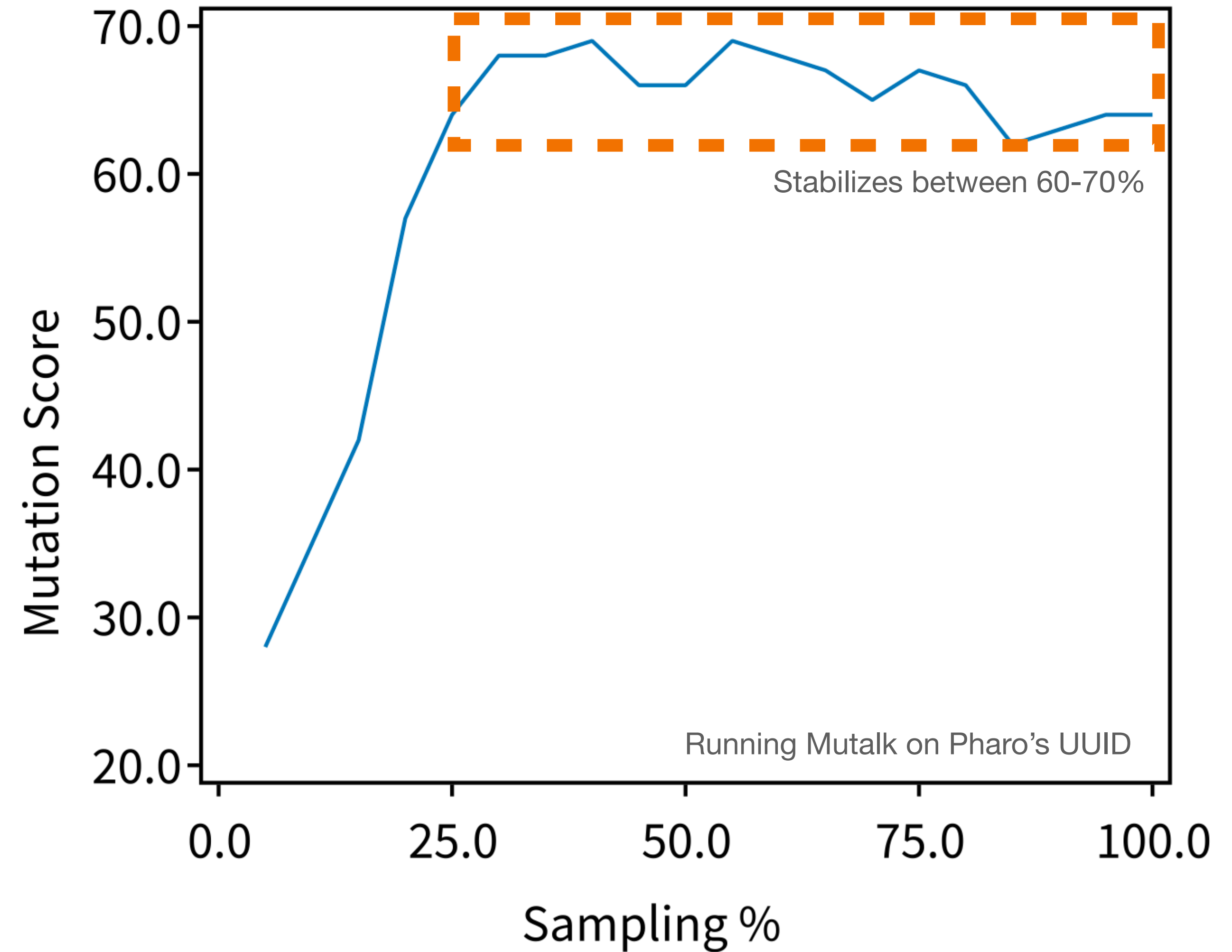
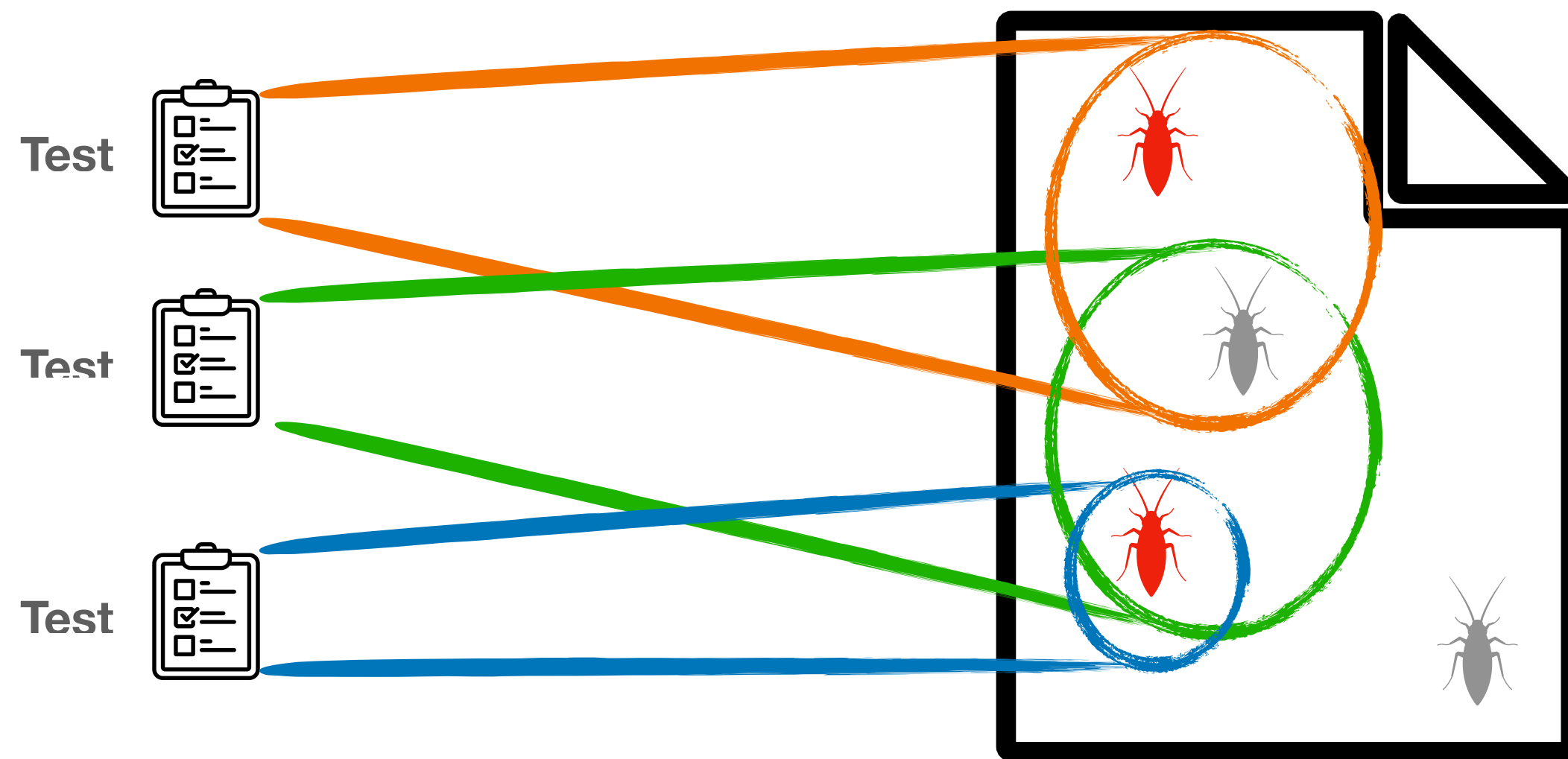
- Run *only* tests covering mutants
- Only mutate covered code



- What if lots of mutants remain?

Random Sampling of Mutants

- Lots on research on it
- Aggressively reduces run time

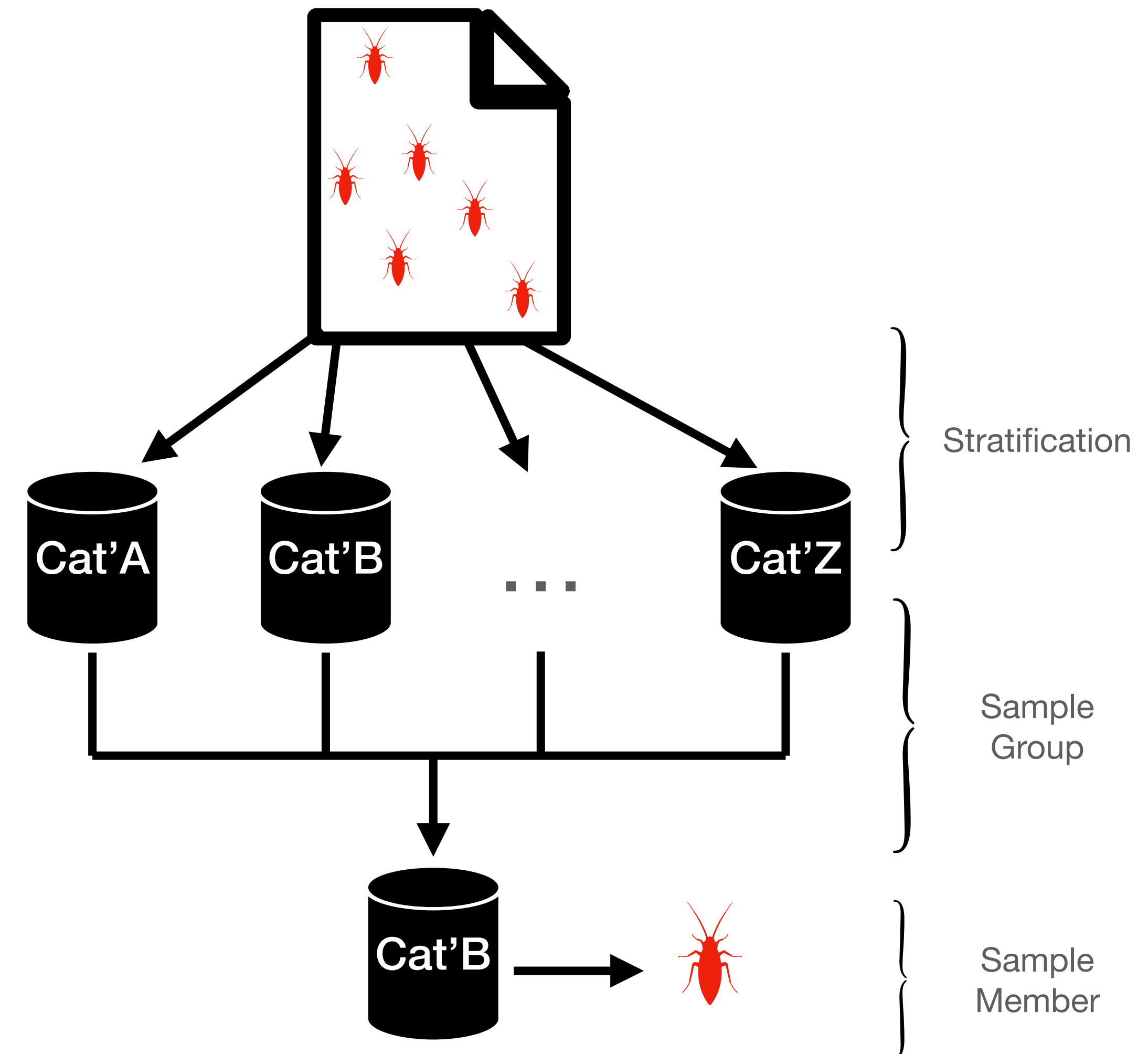


Guiding Random Sampling of Mutants

- Stratified sampling with different strategies
 - per class, method, ...

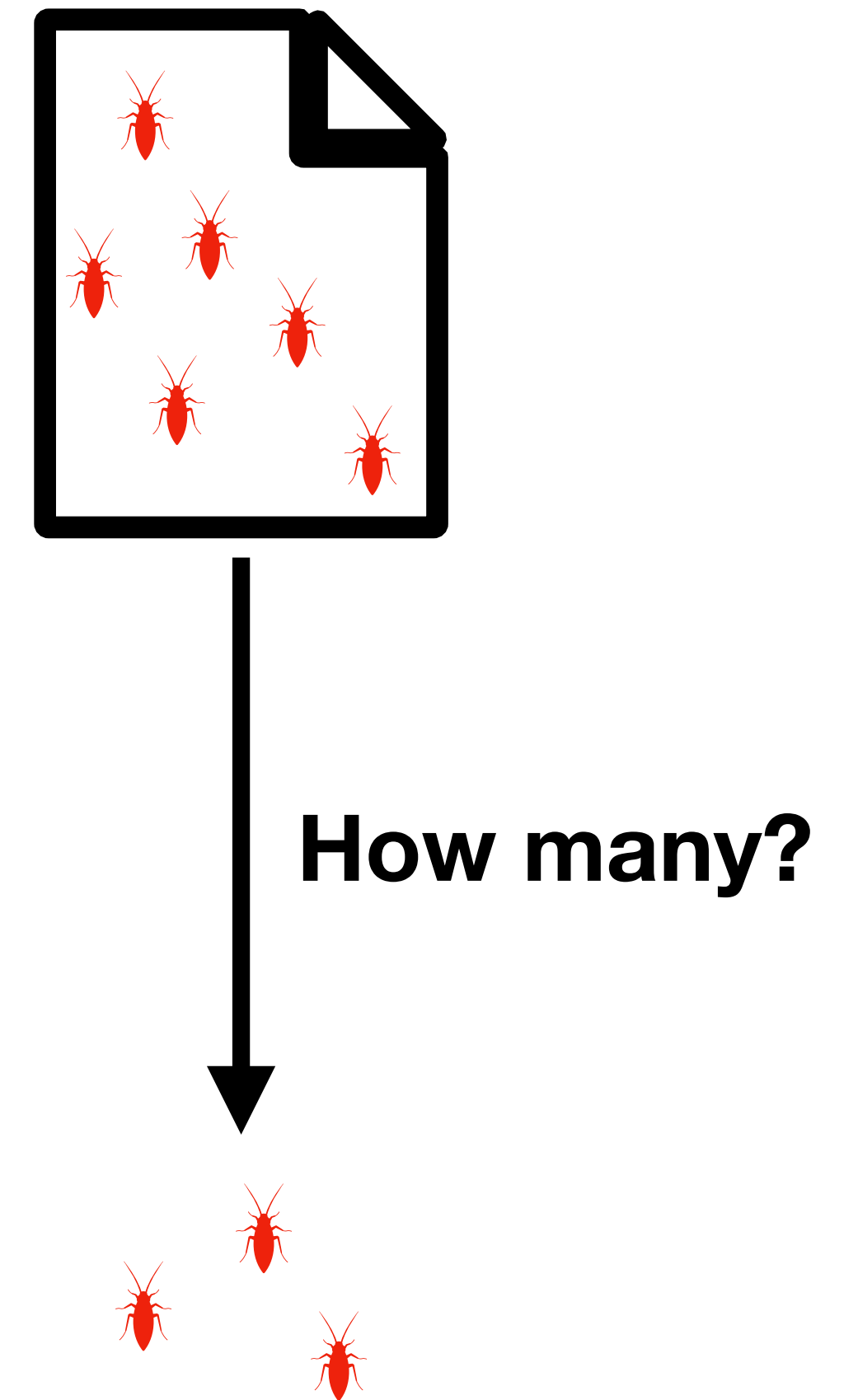
1. Taking a sample:

1. Take a group at random
2. Take a member from the group



When to Stop Sampling: Budgets!

- Fixed budgets: # or % of mutants
 - Too many: No win in runtime!
 - Too few: Unrealistic mutation score...
- We need a practical solution:
 - **Time budget!**



Github actions integration

- Github actions
- + Microdown
- Configure budgets, different run modes...

```
name: MutalkCI

on:
  pull_request_target:
    branches: [ main ]
    paths-ignore:
      - 'README.md'
      - 'resources/**'

jobs:
  mutation_testing:
    uses: pharo-contributions/mutalk/.github/workflows/mutalk-workflow.yml@v2.5.0
```




-  Update PerfTreeParser.class.st
-  Merge branch 'Alamvic:main' into main
-  Merge branch 'Alamvic:main' into main



github-actions bot commented 3 days ago

Mutation results

- Commit: [82aace5](#)

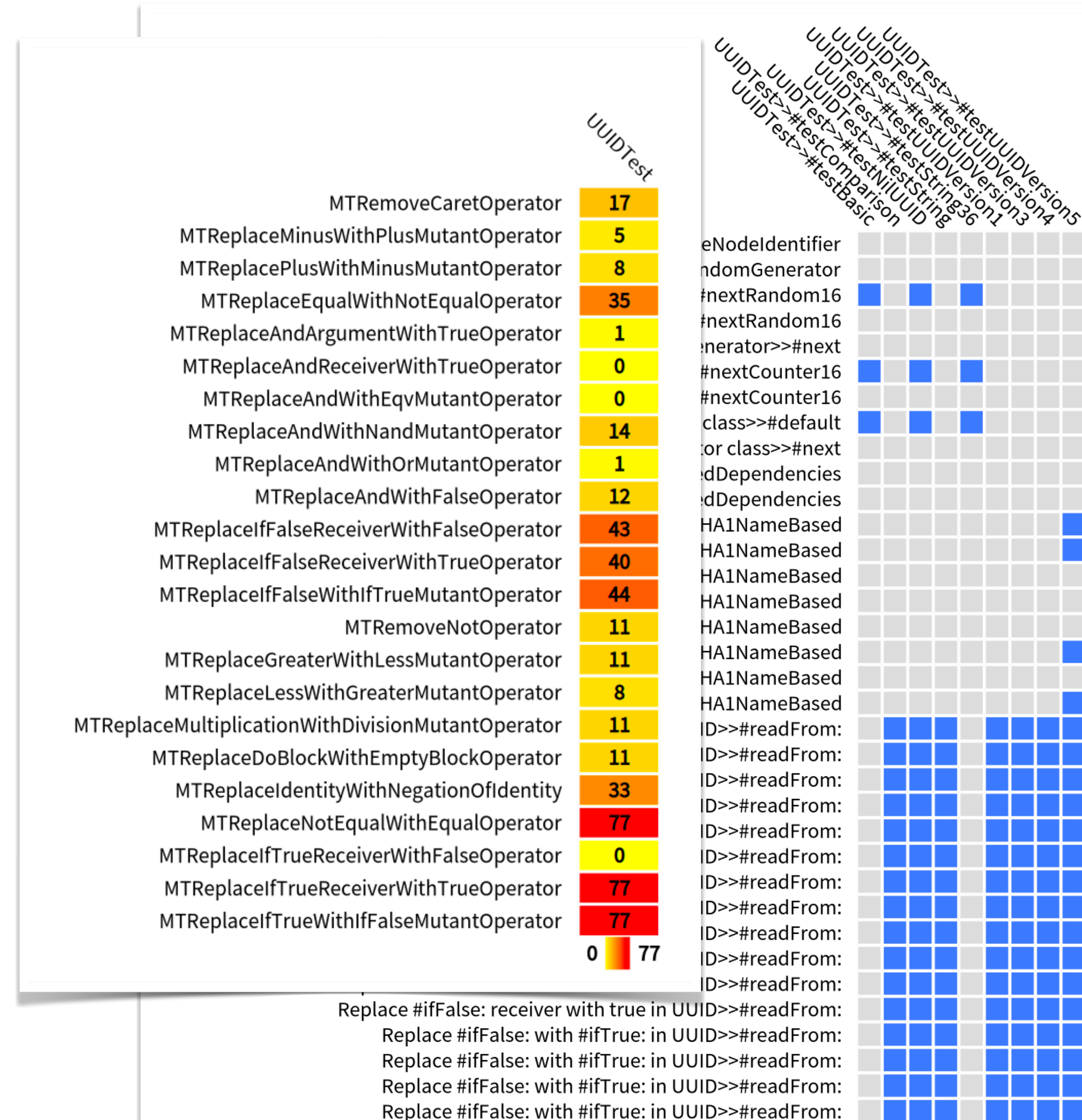
Mutants	Number of Mutants
 Alive	161
 Killed	834
 Terminated	0

Mutation score: 83



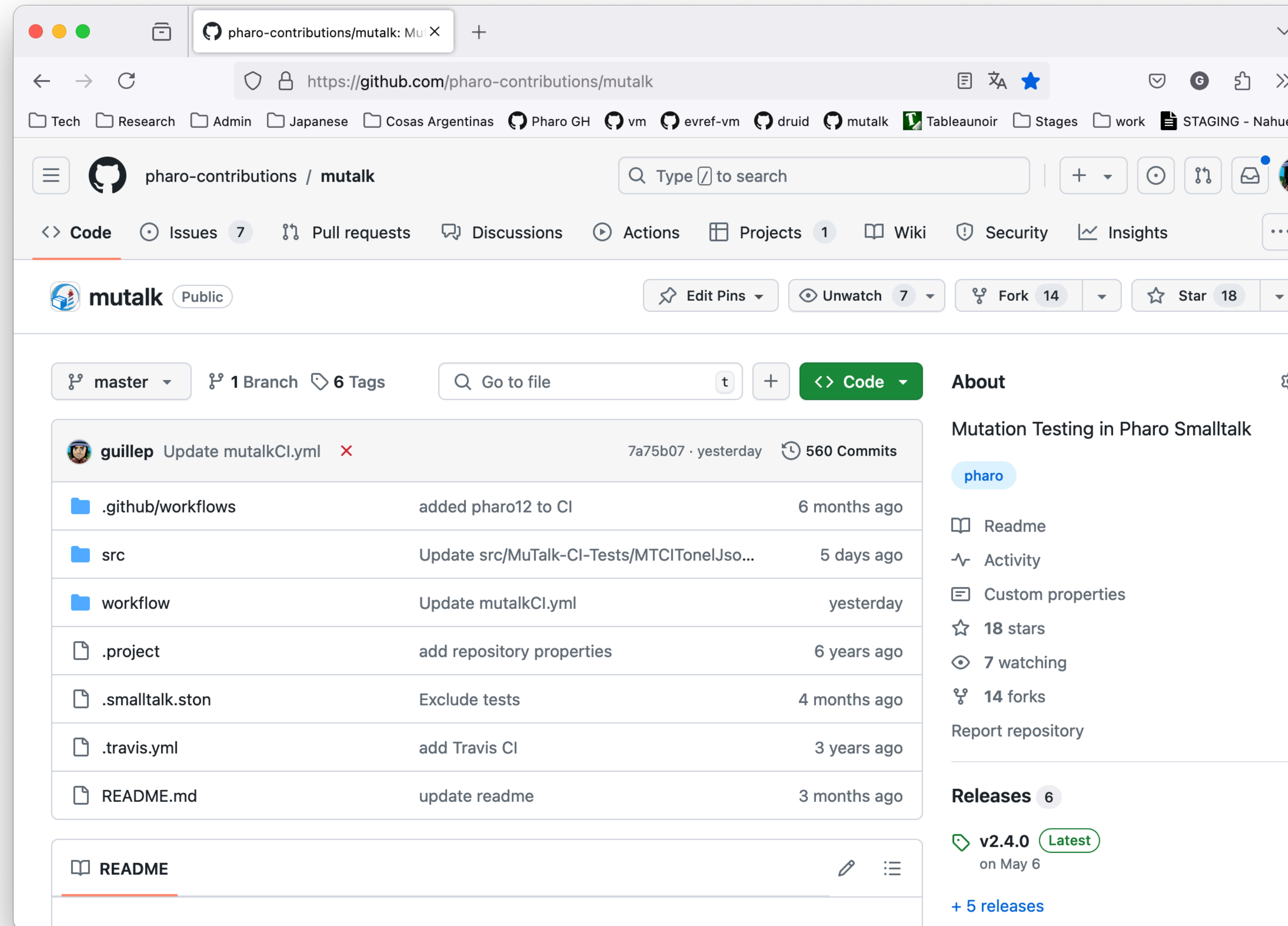
Mutation Matrix and Heatmap

- **Fine-tune** for your project
- **Advanced** mutation analysis
- Identify
 - **Trivial** mutations: easy to kill
 - **Redundant** tests and mutants
 - Randomization strategy



Talk to me

- If you want to **use it!**
- If you want to **contribute!**
- You need **specific operators**



Mutalk 2024

- Working from **Pharo9 to Pharo12 (13?)**
- **Practical:** Budgets, random selection and CI
- **New** features: test filters, logging, new operators
- **Fixes, improvements, cleanups**
- Thanks to all people contributing over the years!

Coverage vs Mutation

- Average has 100% coverage
- What if we mistake the / by a *?

```
CollectionTest >> testAverage  
  self assert: #(1 2 3) average
```

```
Collection >> average  
  ^ self sum / self size
```