

**ARE YOUR TESTS  
GOOD ENOUGH?**

# Reapproach

**Testing strategy**

**Review process**

**QA methodology**

**TDD approach**

**Refactoring routine**

**Michalis Zampetakis**

**Sr Software Enginner**

**ZEDEDA**

**@mzampetakis[.com]**



**SoCraTes Crete**

**dev/staff**  
developers community crete

# Agenda

- **Code validation**
- **Code coverage**
- **Code mutation**
- **Mutation testing**
- **Challenges**
- **When and how (not) to use it**
- **Key takeaways**

> This talk contains personal views and opinions

How do we **validate** our code?

The code does what it is supposed to do

The code doesn't do what it is not supposed to do

**Tests**



**How do we assess the quality of our tests?**

**Review our code and tests**

**All wrote code AND tests**

**We are all rock-star developers**

**We do TDD**

# Code Coverage



**Code coverage is a percentage measure of the degree to which the source code of a program is executed when a particular test suite is run.**

# Code Coverage



- Line
- Branch
- Functions
- Path
- Loop
- Condition



**DEMO TIME**



**Executing** code and **testing** code  
are not the same thing

**"Program testing can be used to show the presence of bugs, but never to show their absence!"**

**> Edsger W. Dijkstra**

**Code coverage tells us what**

**has not been tested**

**{ When a measure becomes a target,  
it ceases to be a good measure. }**

**> Goodhart's law**

If you want to know whether your test suite works, introduce a **bug** into the code

Then, see if your tests can find it



**DEMO TIME**



**A small modification to our code is called a **mutation** and the generated code is called a **mutant****

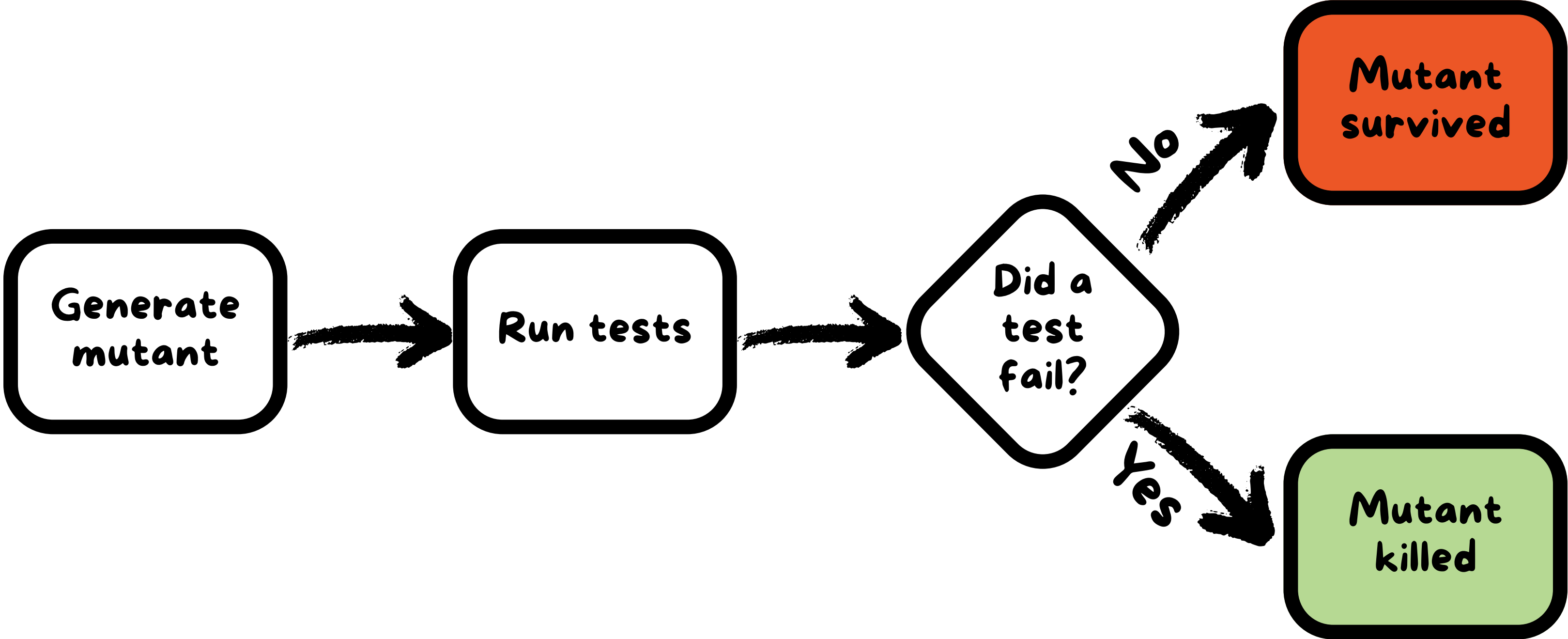
If a mutant does not cause any test to fail, it **survives**

If a mutant does cause a test to fail, it is **killed**

# Mutation operations

>=	>
+	*
0	!
"fizzbuzz"	""
foo()	bar()
foo()	//foo()
if A    B	if false    B
if condition	if !condition

**Automate** and **repeat** it through  
a consistent process



$$\text{mutation score} = \frac{\text{mutants killed}}{\text{mutants generated}}$$



**DEMO TIME**



Code coverage tells us what

**has not** been tested

Mutation testing tells us which

code is **actually** tested

**Does this  actually work?**


# Coupling effect



- Explicitly testing simple faults
- Implicitly testing for more complicated ones
- Minor bugs can have significant impacts

**Mutation testing challenges**

# Equivalent Mutants






**DEMO TIME**



# Equivalent Mutants



- Fix the test
- Redundant code
- Refactor the code

**Performance**



**[github.com/sirupsen/logrus](https://github.com/sirupsen/logrus)**

**2.7" testing time**

**80% code coverage**

**200 mutants**

**9' mutation testing time**

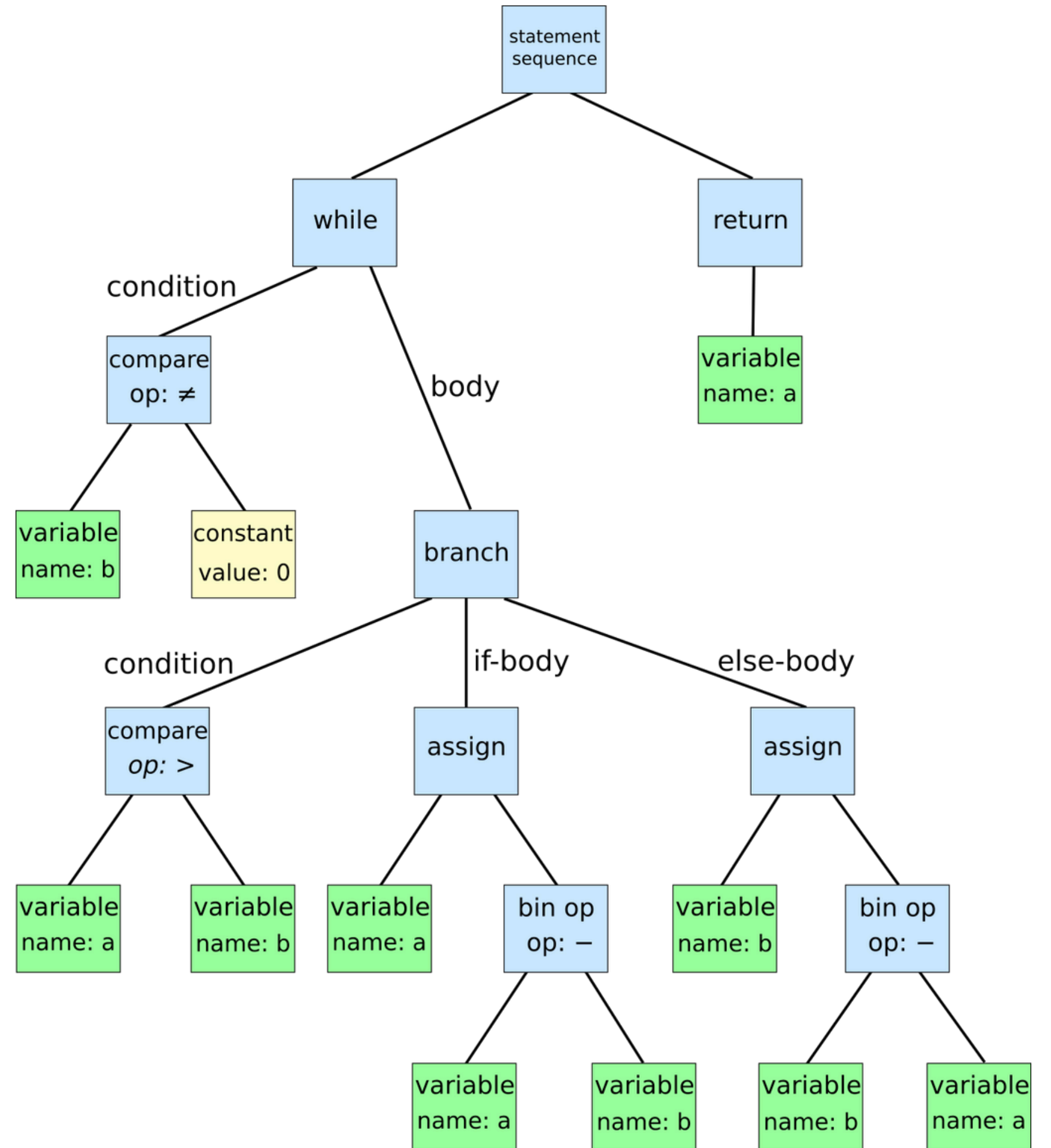
# Optimizations

- **Multi-core/threads**
- **Kill fast**
- **Run cheap tests first**
- **Test covered mutation**
- **No compilation cycles**

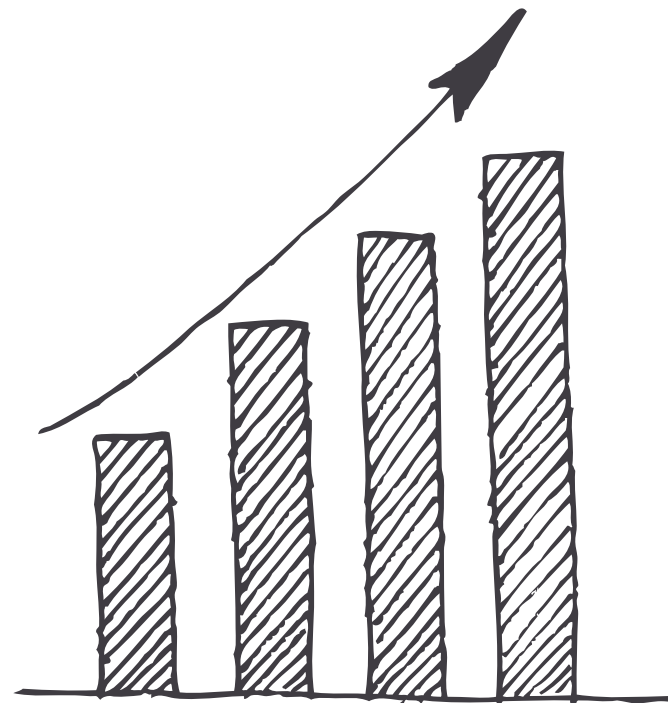
```

while b ≠ 0:
  if a > b:
    a := a - b
  else:
    b := b - a
return a

```



What about our **huge** codebases?



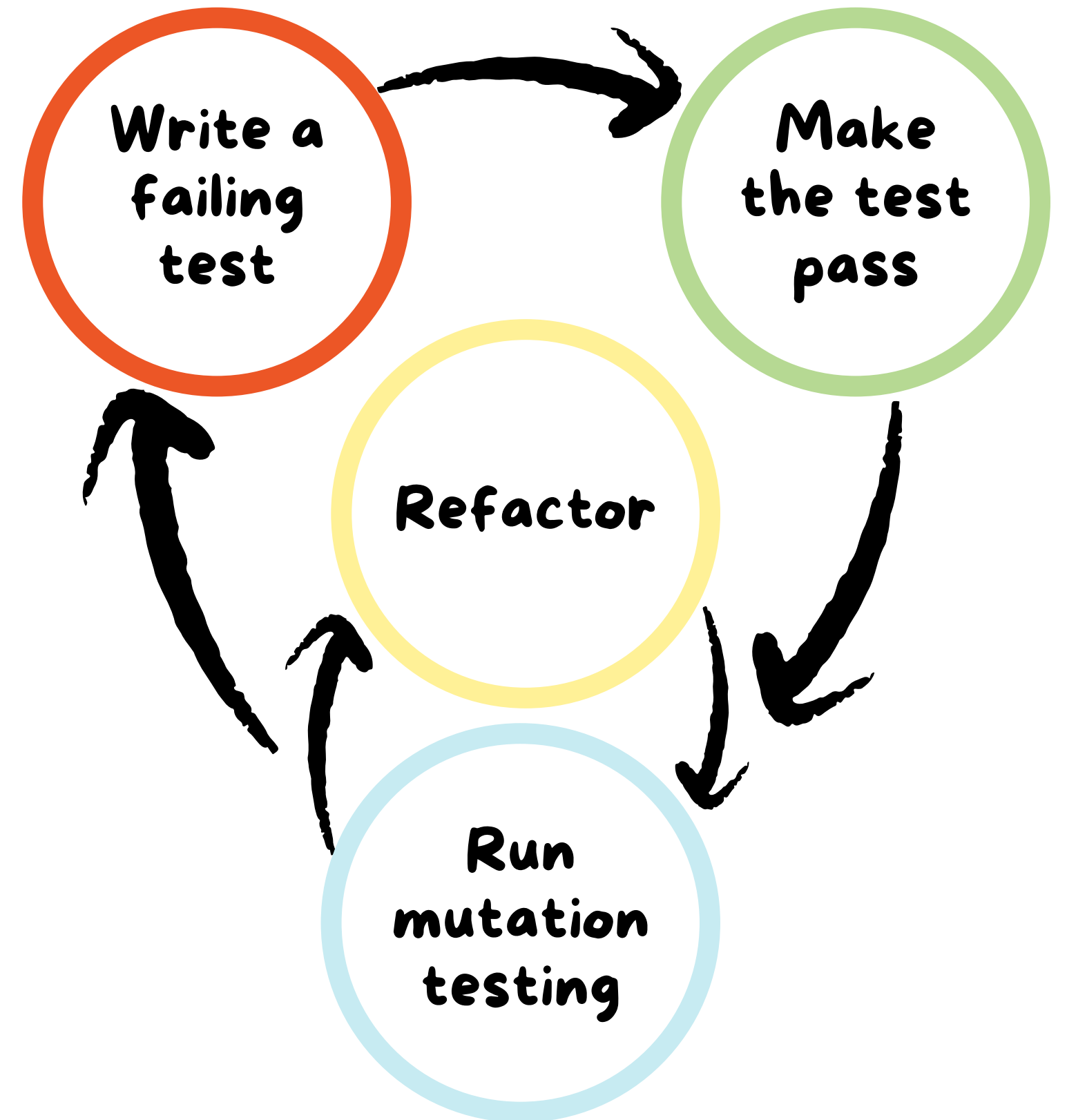
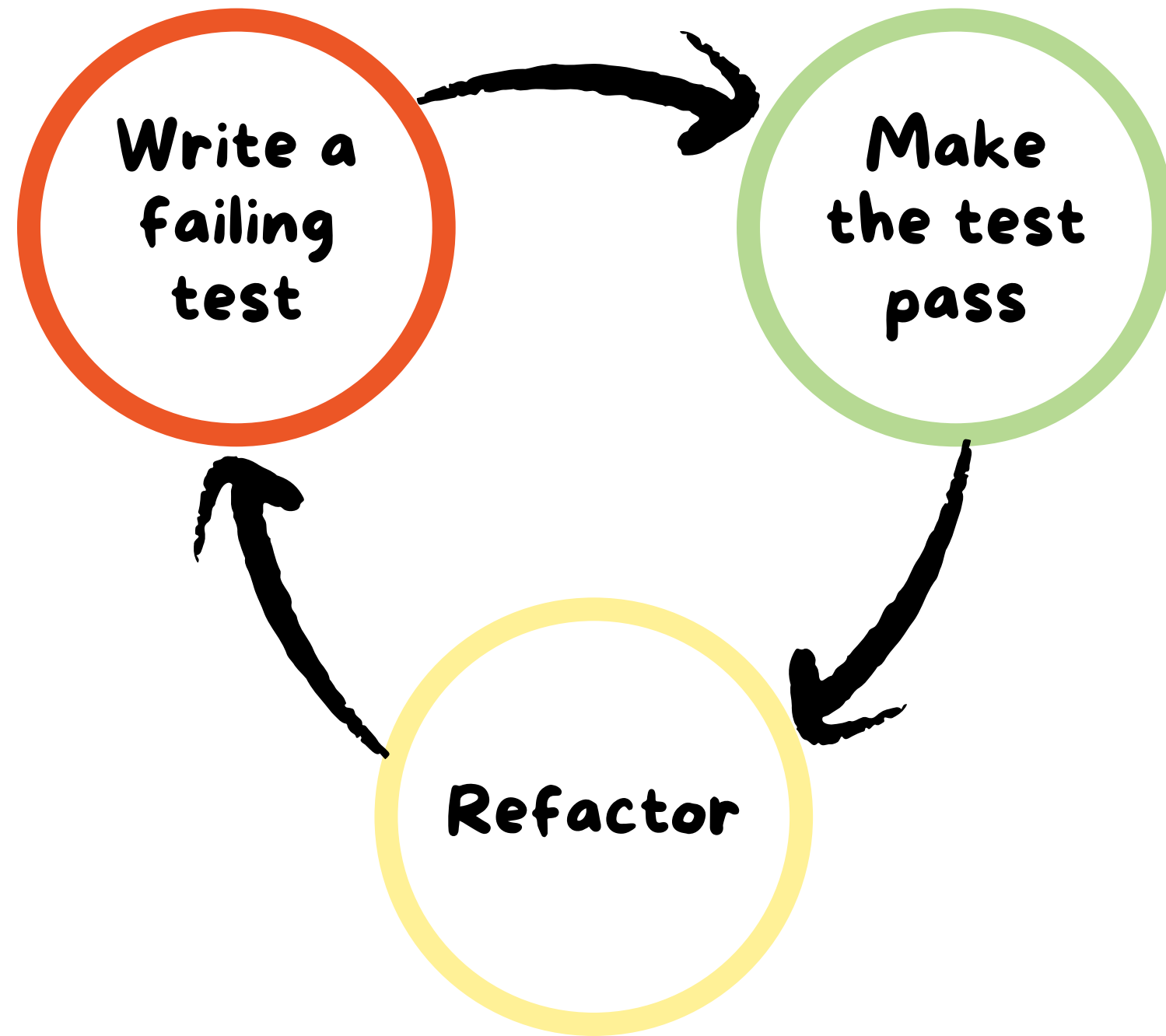
# **Techniques**

- **Exclude code to be mutated**
- **Exclude specific mutants**
- **Avoid extreme mutations**
- **Run mutation testing periodically**

**Doesn't matter**



# TDD



# **Methodologies**

- **Analyze a small piece of code**
- **Integrate with VCS**
- **Run on critical parts**
- **Refactor complex (untested) code**

**Don't care about the  score**

It's a **tool** not a number

# Key takeaways

- **Code Coverage  $\neq$  test coverage**
- **Write tests for a reason**
- **Choose test inputs wisely**
- **Aim for high code coverage**
- **Choose the mutation library**

# Key takeaways

- **Mutation Testing**
  - **Completes our test suite**
  - **Finds missing tests**
  - **Finds incomplete tests**
  - **Serves as a safety net for refactoring our code & tests**
  - **Finds redundant code & code smells**
  - **Reapproach the development process**

# References



- <https://github.com/avito-tech/go-mutesting>
- <https://mzampetakis.com/posts/Testing-the-Tests>
- <https://github.com/theofidry/awesome-mutation-testing>
- [https://en.wikipedia.org/wiki/Abstract\\_syntax\\_tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree)
- <https://www.thoughtworks.com/insights/podcasts/technology-podcasts/better-testing-through-mutations>
- <https://blog.cleancoder.com/uncle-bob/2016/06/10/MutationTesting.html>

**ARE YOUR TESTS GOOD ENOUGH?**

**THANK YOU!**



**Michalis Zampetakis**

**Voxxed Days  
Crete 2025**