A STORY OF MUTANTS

Who watches the watchers?

Isabel Garrido

Senior backend developer

SEAT CODE

letgo

step 4ward

BCN ENGINEERING

JOIN NOW

{▸}CodelyTV

@isabeliita90

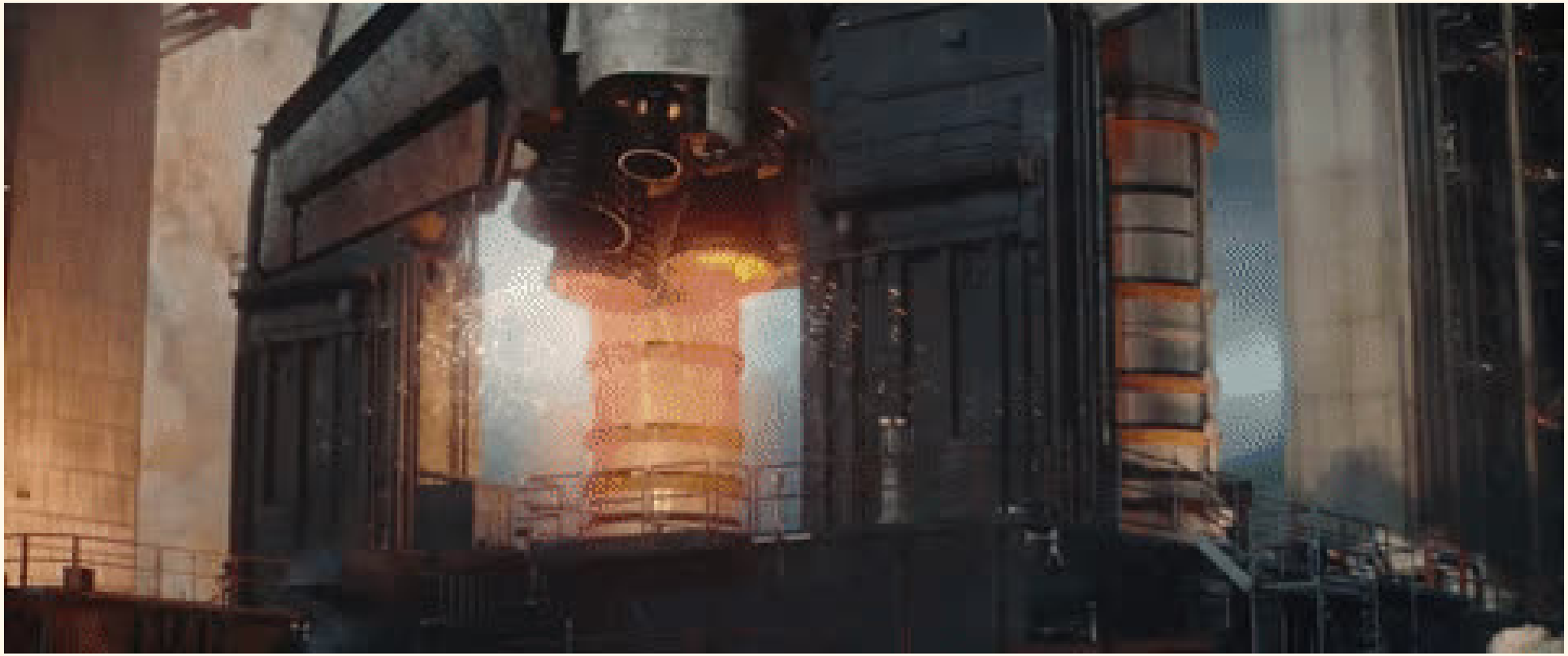
STRUCTURE

- Once upon a time...
- What is mutation testing?
- Show me the code

ONCE UPON A TIME...

...THERE WAS A PROJECT...

...AND A TEAM



AND THE NIGHTMARE BEGAN

When a metric becomes a target you fool it.

GOODHART'S LAW

Code Coverage

Number of Classes		Line Coverage
392	67%	4929/7386

Code Coverage

Number of Classes		Line Coverage
392	67%	4929/7386

Mutation testing

Mutation Coverage	
23%	2477/10815

MUTATION TESTING

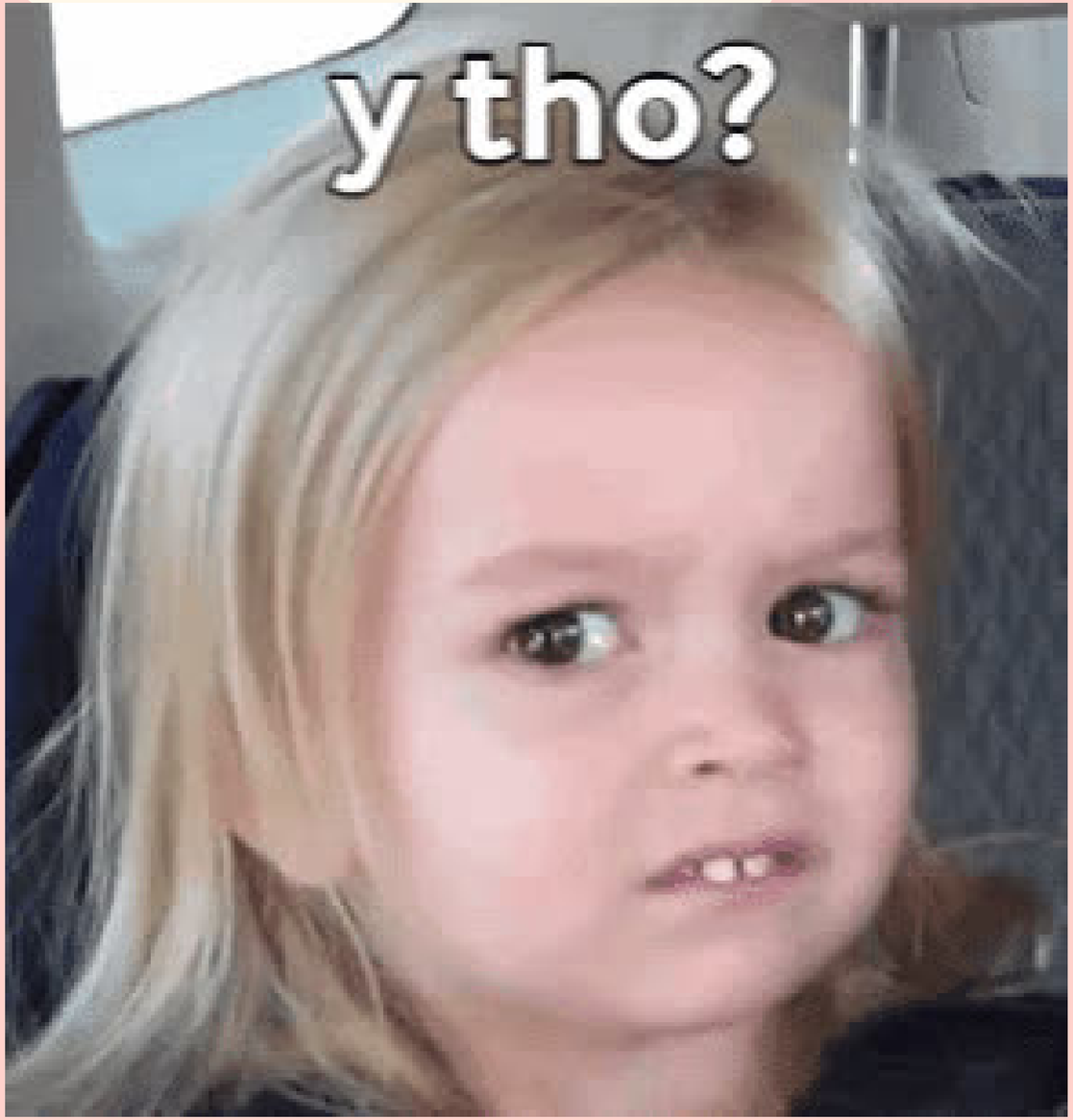
MUTATION

TO KILL A MUTANT

- **R** *Reachability*
- **I** *Infection*
- **P** *Propagation*

BASES

- competent programmer hypothesis
- coupling effect


y tho?

Tests can be created to verify the correctness of the implementation of a given software system, but the creation of tests still poses the question of whether the tests are correct and sufficiently cover the requirements that have originated the implementation.

WIKIPEDIA

EXTREME MUTATION TESTING

Extreme mutation testing in practice: An industrial case study

SHOW ME THE CODE

```
plugins { this: PluginDependenciesSpecScope
    id("info.solidsoft.pitest") version "1.7.0"
}
```

```
pitest { this: PitestPluginExtension
    setProperty("junit5PluginVersion", "0.12")
    setProperty("testPlugin", "junit5")
    setProperty("targetClasses", listOf("org.review_algorithms.*"))
    setProperty("outputFormats", listOf("HTML"))
    setProperty("threads", 2)
    setProperty("withHistory", true)
}
```

```
./gradlew pitest
```

github-actions bot commented 13 seconds ago

🧛 Mutation Testing Summary 🧛

Line Coverage: 26/26 (100%)
Generated 15 mutations Killed 15 (100%)
Mutations with no coverage 0. Test strength 100%
Ran 17 tests (1.13 tests per mutation)

Check artifacts for the complete report

RECAP

- Tests are code so they can be not correct or insufficient
- Mutation testing is a technique to know the reliability of our test suit
- Code coverage is way faster and requires fewer resources but it's easy to trick
- It's more useful to introduce mutation testing on a regular base development process than as a one-time occurrence

RESOURCES

- Extreme mutation testing in practice: An industrial case study
- Suggestions on Test Suite Improvements with Automatic Infection and Propagation Analysis
- Descartes: a PITest engine to detect pseudo-tested methods - Tool Demonstration
- Domain-RIP Analysis: A Technique for Analyzing Mutation Stubbornness
- Mutation 2000: Uniting the Orthogonal
- MuJava : An Automated Class Mutation System
- The Fallacy of the 100% Code Coverage
- Pitest documentation

THANK YOU!