

Mutation testing

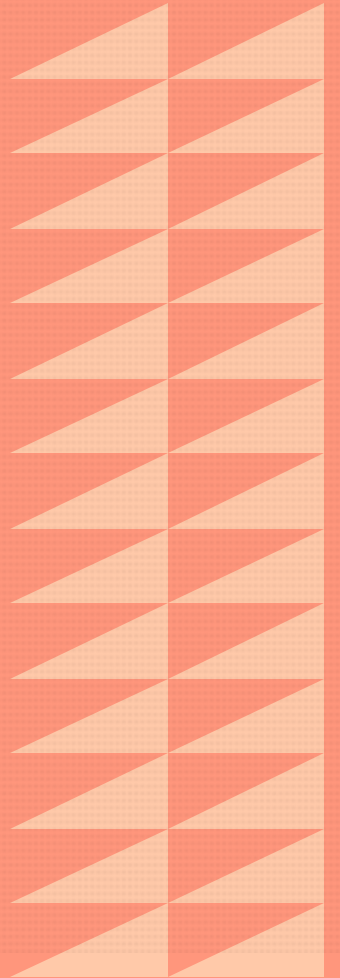
About the idea and constraints
in commercial use

Wojtek



00

**Buzzword or
salvation?**





*„Mutation testing reveals weak spots in test suites, improving error detection by **30%**.”¹*



*„Systems tested with mutation have shown to be **25%** more resilient to unforeseen real-world scenarios.”²*



*„Projects using mutation testing report a **20%** decrease in post-release defects.”³*

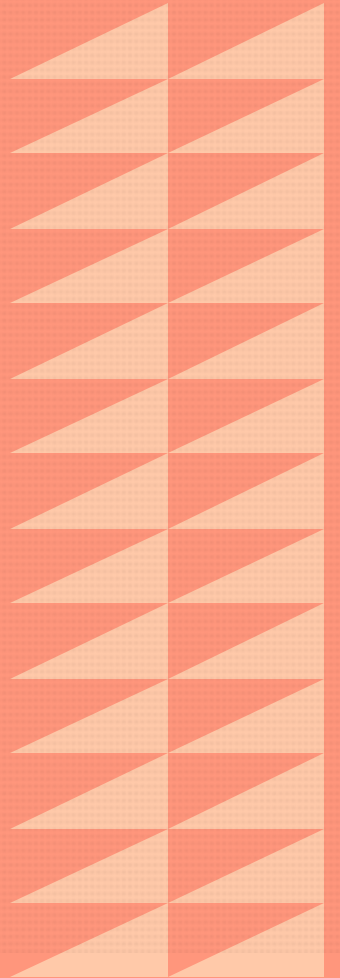


*„The tests written with the TDD+M approach achieve **17%** better statement coverage and **23%** better mutation coverage than the tests written with the TDD approach.”⁴*

1. A. J. Offutt, "Mutation Testing and the Use of Software Test Data Sets"
2. L. Madeyski, "Effectiveness of Mutation Testing: Experimental Evaluation with Real Software" (PWr)
3. Y. Jia and M. Harman, "An Analysis and Survey of the Development of Mutation Testing"
4. A. Roman and M. Mních, "Test-driven development with mutation testing – an experimental study" (UJ)

01

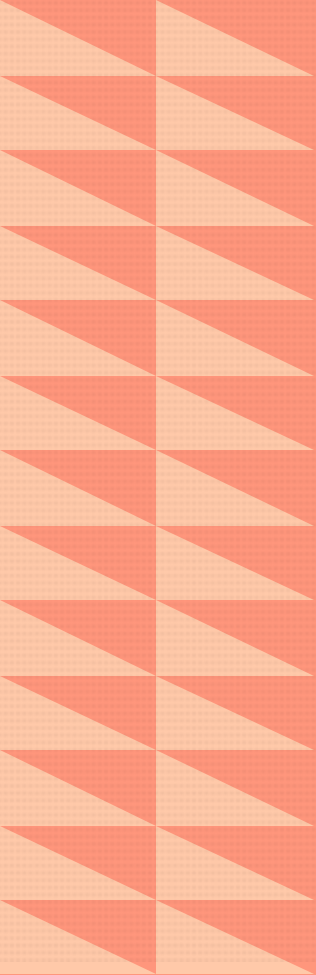
Introduction to mutation testing



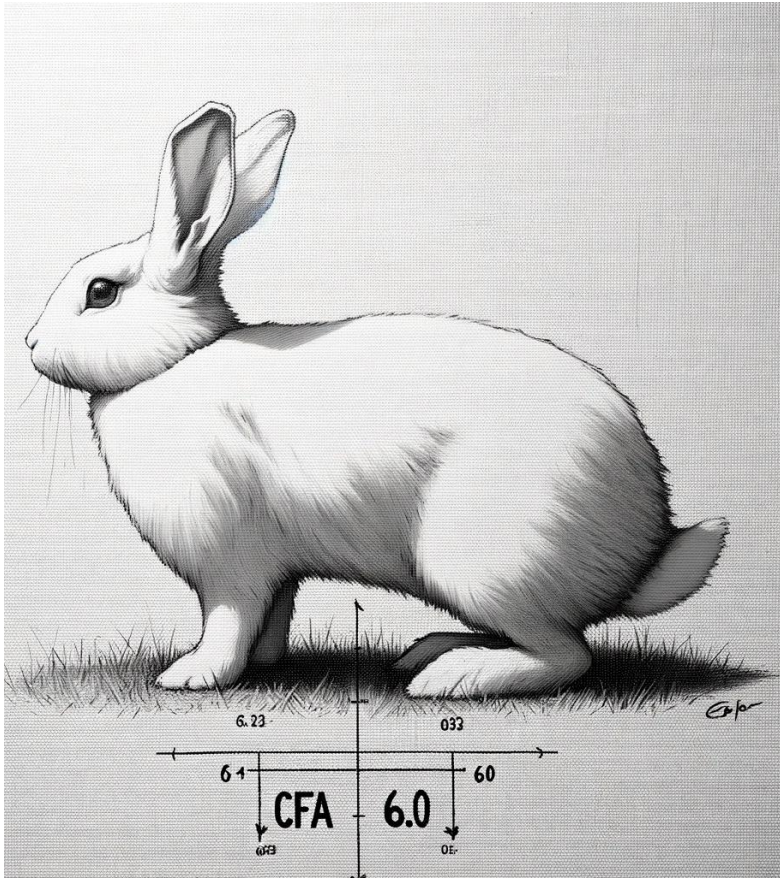
Mutation

Mutation



A decorative graphic on the left side of the slide, consisting of two vertical columns of triangles. Each column contains 12 triangles, for a total of 24. The triangles are arranged in a staggered pattern, with the top triangle of the right column offset to the right relative to the top triangle of the left column. The triangles alternate in color between a light orange and a slightly darker shade of orange.

„Unexpected change or
modification in a given
structure/system/pattern,
that can lead to new
behaviors.“



```
public class Rabbit {  
  
    public static boolean isHungry(int foodEaten) {  
        return foodEaten <= 3;  
    }  
}
```



```
public class RabbitTest {  
  
    @Test  
    public void testIsHungryWhenSatietyLevelIsThree() {  
  
        var rabbit = new Rabbit();  
        int foodEaten = 3;  
  
        assertTrue(rabbit.isHungry(foodEaten));  
    }  
}
```


After a mutation...



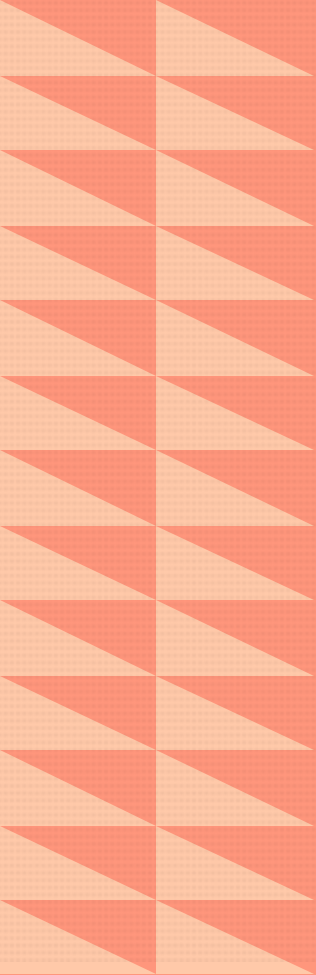
```
public class Rabbit {  
  
    public static boolean isHungry(int foodEaten) {  
  
        // mutation: from <= to >=  
        return foodEaten >= 3;  
    }  
}
```



```
public class RabbitTest {  
  
    @Test  
    public void testIsHungryWhenSatietyLevelIsThree() {  
  
        var rabbit = new Rabbit();  
        int foodEaten = 3;  
  
        assertTrue(rabbit.isHungry(foodEaten));  
    }  
}
```



**How is mutation
defined within the
application lifecycle?**



The more code
you write, the
more new bugs
you create.

Zhang, Hongyu. (2009). An
Investigation of the Relationships
between Lines of Code and Defects.

**A mutation is a
simulated potencial
programmer mistake.**

It doesn't concern us – but are we sure?

```
def isExpired(reservation: WaitingReservation): Boolean = {  
  val currentTimeInMillis = new Date().getTime  
  val reservationOpenTimeInMillis = reservation.date.getTime  
  val expirationTimeInMillis = minExpirationTime.toMillis  
  
  reservationOpenTimeInMillis + expirationTimeInMillis <= currentTimeInMillis  
}
```

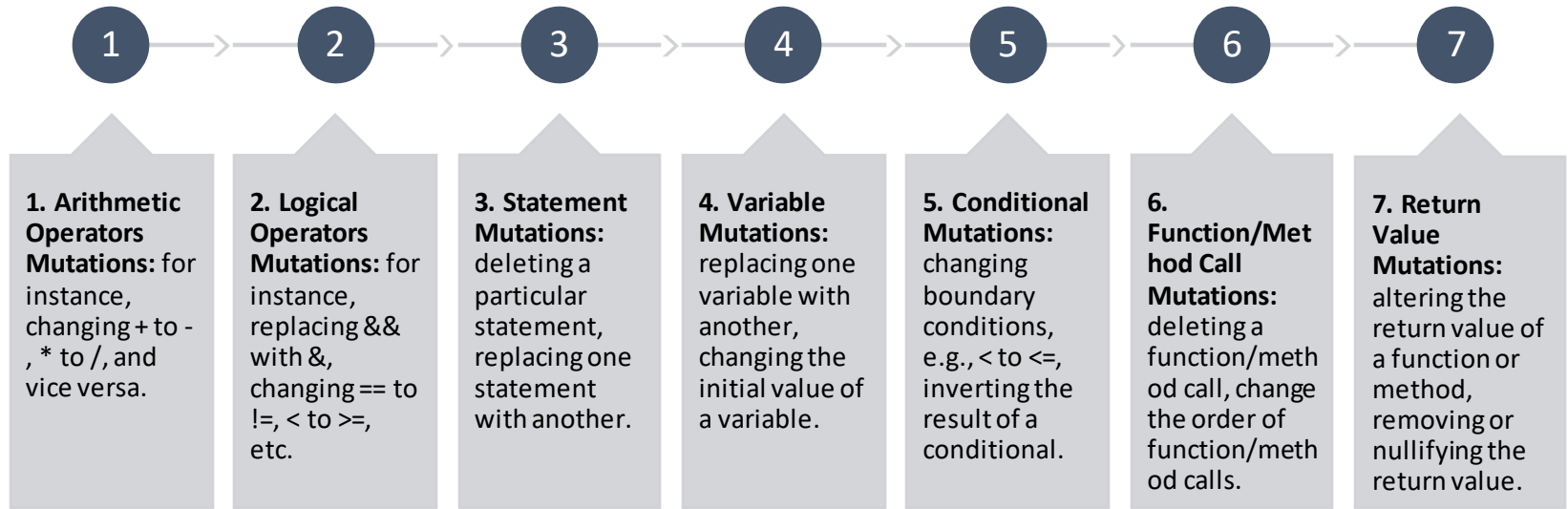


It doesn't concern us – but are we sure?

```
def isExpired(reservation: WaitingReservation): Boolean = {  
  val currentTimeInMillis = new Date().getTime  
  val reservationOpenTimeInMillis = reservation.date.getTime  
  val expirationTimeInMillis = minExpirationTime.toMillis  
  
  reservationOpenTimeInMillis + expirationTimeInMillis > = currentTimeInMillis  
}
```



Operators mutants



And...

... many more!

Operators mutants

- In general: replace any operator with its opposite

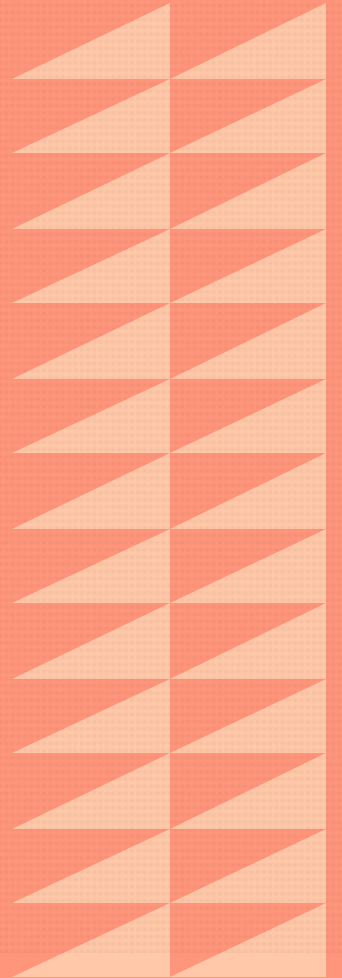
**As many operators as
you can define.**

Terminology

- *Mutation, mutant*
- *Mutant survival, killing a mutant*
- *Mutation score = killed mutants / all mutants*

02

Mutation testing for Java and Scala



Popular Java libraries

- PIT (Pitest)
- MuJava
- Javalanche
- Jester

Scala?



Scala?

- Stryker4s
- PIT (Pitest) - why not:



Henry Coles <henry.coles@googlemail.com>

Do: Ty

DW: hello@henrycoles.com

Unfortunately PIT's approach to mutation testing is not a good fit for Scala. PIT mutates jvm bytecode, which for Java and Kotlin maps back to the source in a fairly straightforward way. For scala, the mapping is much more complex. When I last looked at this (quite some time ago) the bytecode constructs the compiler generates for each language feature also changed greatly with each scala release.



Śr, 26.07.2023 23:12

Pitest

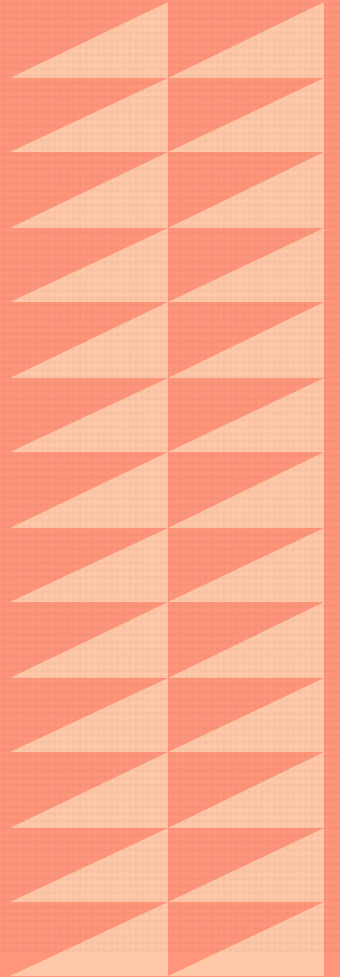
- Java, Kotlin and... Scala (?)
- Gradle, Maven, Ant, command line
- allows configuring multiple settings
- mutation optimizations

Pitest

- <https://github.com/wszlosek/mutation-testing-xref.git>

03

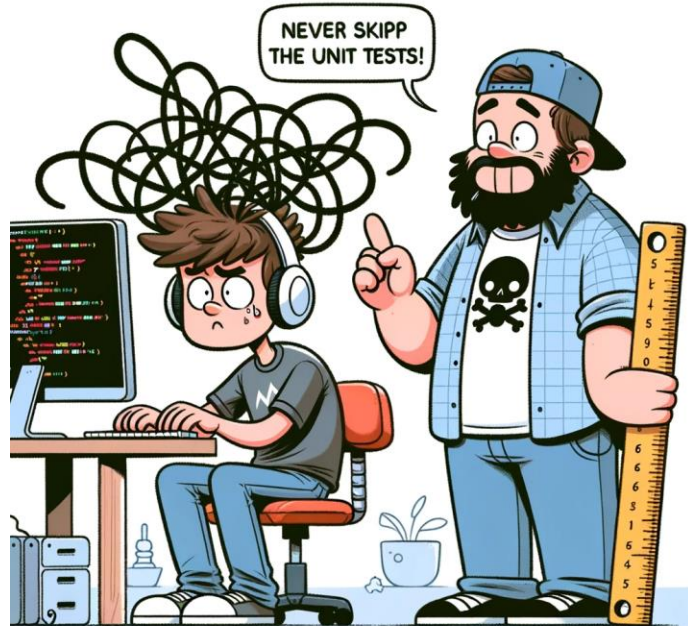
**Mutation testing
in the commercial
context**



**Why don't most*
companies use
mutation testing?**

* - 58% (*State of Software Testing 2021*)

Well-maintained unit tests (must have)



Optimization, Stupid!

Let:

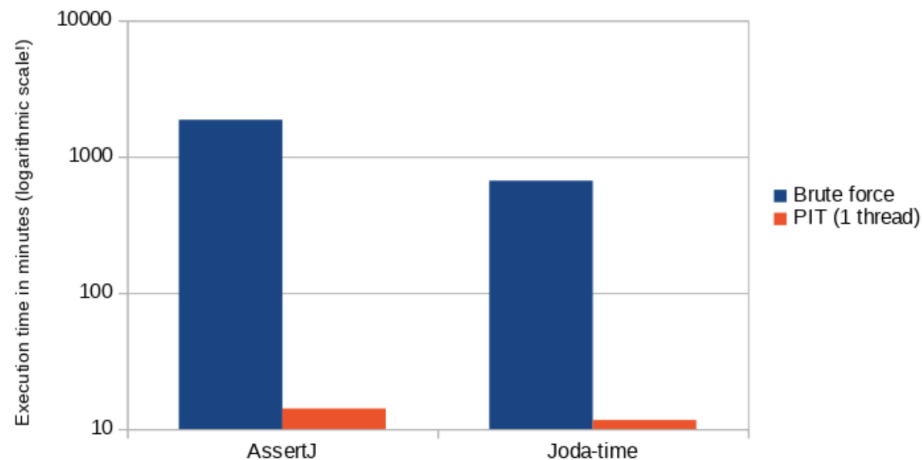
- $n = \text{\#tests}$
- $m = \text{\#mutants_types}$
- $k(m) = \text{\#places_suitable_for_mutation_m}$

$n * m * k(m)$ [operations] (brute-force)

But...

EXECUTION TIME IN MINUTES	BRUTE FORCE	PIT (1 THREAD)
AssertJ	1866.67	14.15
Joda-time	666.67	11.65

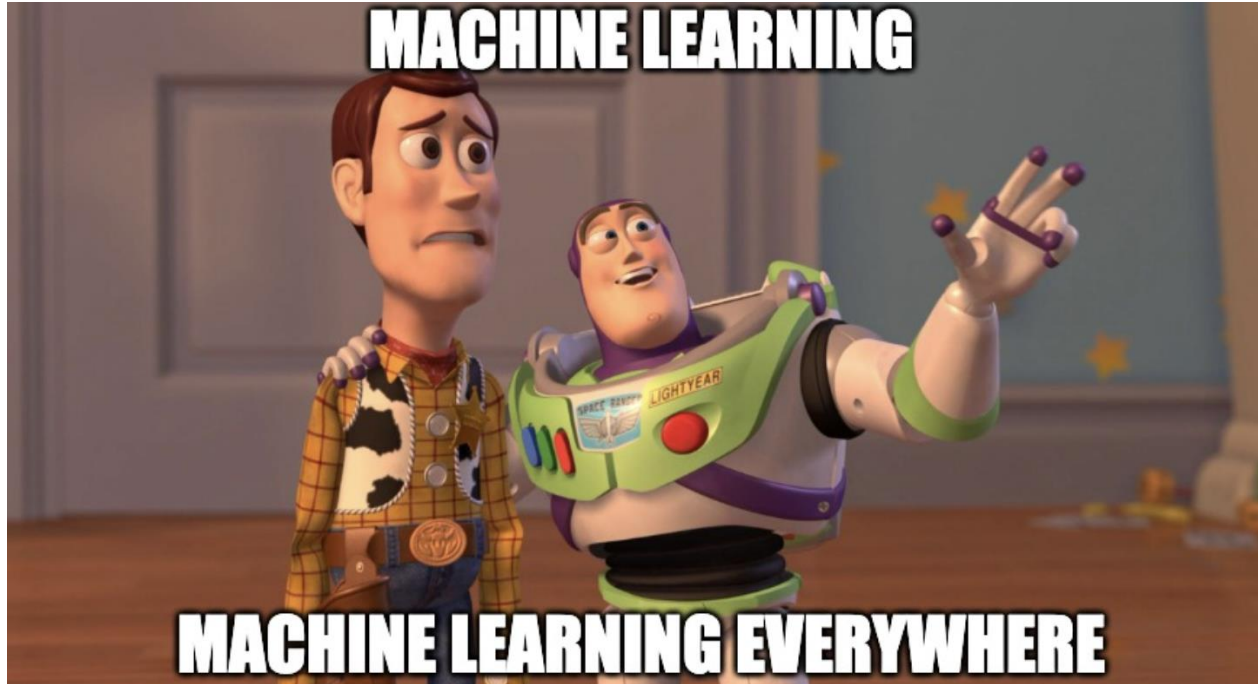
Brute force vs PIT - mutation testing execution time



Methods of optimization include:

- randomly picking mutations
- utilizing results from previous tests
- mutating only the most important parts of the code

And, of course!



So, will mutation testing ever become a popular practice in many companies?

Questions?

