

# Wir testen. Aber testen wir auch gut genug?

Eine Einführung in Mutation Testing mit Stryker.NET

Patrick Drechsler



# Patrick Drechsler

- Software Entwickler
- Beruflich: C#
- Interessen:
  - Software Crafting
  - Test-Driven Development
  - Funktionale Programmierung
  - Domain-Driven Design
- Slides sind online: Siehe QR-Code

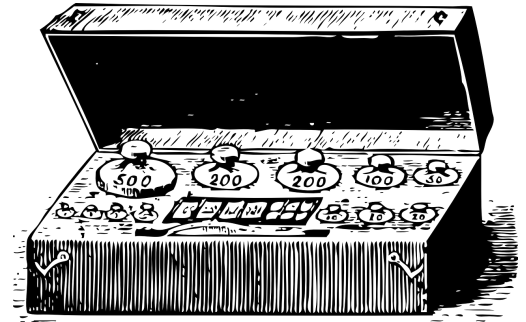




## Let's talk about "Metrics"...

<https://www.nngroup.com/articles/campbells-law/>

- "It is wrong to suppose that if you can't measure it, you can't manage it - **a costly myth.**" - W. Edwards Deming
- **Campbell's Law** states that **the more important a metric is** in social decision making, the **more likely it is to be manipulated.**
- **Goodhart's Law** states that **"When a measure becomes a target, it ceases to be a good measure"**





# Test coverage

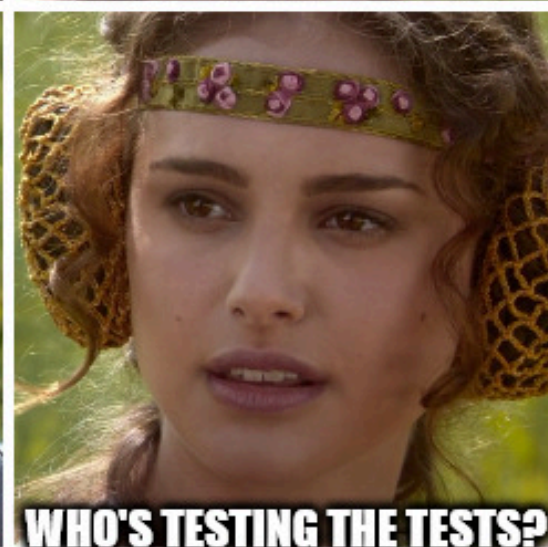
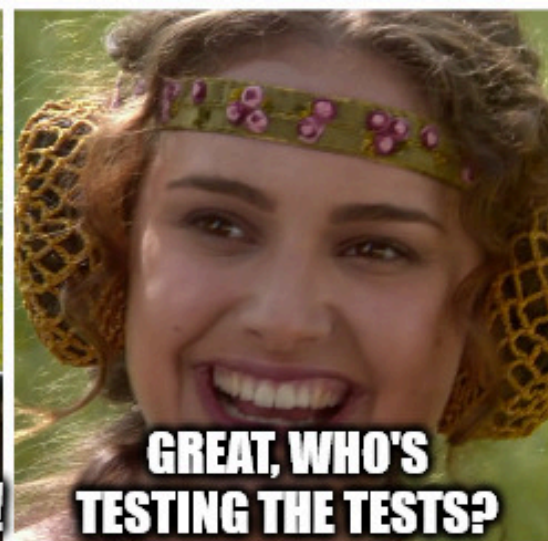
- 🎓 defines the percentage of covered code
- ✅ 100% test coverage means, every line of code is executed at least once
- ⚠️ 100% test coverage **does not mean that every scenario / use-case is covered**



# Is Test coverage a "good metric"?

- not every line of code needs to be tested
- BUT: having no tests is obviously also not a good idea
- anything above 60% is a good baseline (but, "it depends")
- test coverage does not tell us anything about the **quality** of the tests







# What is mutation testing?

<https://stryker-mutator.io/docs/>

- Mutation testing **introduces changes to your code**, then runs your unit tests against **the changed code**.
- the "change" is called a **mutant**
- If our test suite is ok for a "mutant:" Ups, we missed something

# Hello-World Example

Production code:

```
public string DoMagic(int i) => i < 18 ? "child" : "adult"
```

- `dotnet stryker`
- it creates a mutant replacing `<` with `<=`

```
public string DoMagic(int i) => i ≤ 18 ? "child" : "adult"
```

- The mutant "survived"
- The mutant did not provoke a test failure!
- ⚠ Our test suite might not be good enough! ⚠

Test suite (100% code coverage!):

```
[Theory]
[InlineData(10, "child")]
[InlineData(20, "adult")]
public void DoMagic_works(int input, string expected)
{
    DoMagic(input).Should().Be(expected)
}
```







# Mutations

Let's have a look at mutations:

<https://stryker-mutator.io/docs/stryker-net/mutations/>

Most mutations are language agnostic

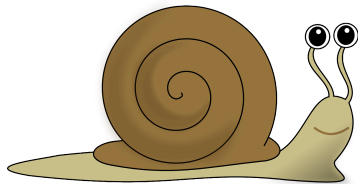
Some are optimized for .NET:

- [Initializers](#)
- [Removal](#)
- [Linq](#)
- [Null-coalescing Operators](#)





## Isn't this slow?



- Short answer: YES
- BUT: **These frameworks have smart heuristics for short circuiting**
- CI: Don't include this in normal commits
- CI: use "Nightly", or local (for **exploratory analysis**)
- Google uses Mutation Testing on really large projects: <https://research.google/pubs/practical-mutation-testing-at-scale-a-view-from-google/>
  - "... a codebase of **two billion lines of code** and more than **150,000,000 tests**"
  - "... used by more than **24,000 developers** on more than **1,000 projects**"
  - It is still slow, but not as slow as you might think



# Mutation Strategies

🤔 How do these frameworks optimize performance?

<https://stryker-mutator.io/docs/stryker-net/technical-reference/research/#comparison>

- mutate **source code**
  - mutate **byte code**
  - **mutant schemata** (aka "mutant switching")
- 👉 Stryker.NET uses "mutant schemata"





# Live coding





# Reports: HTML (Overview)

## All files Stryker.NET Report



Mutants Tests

All files

30

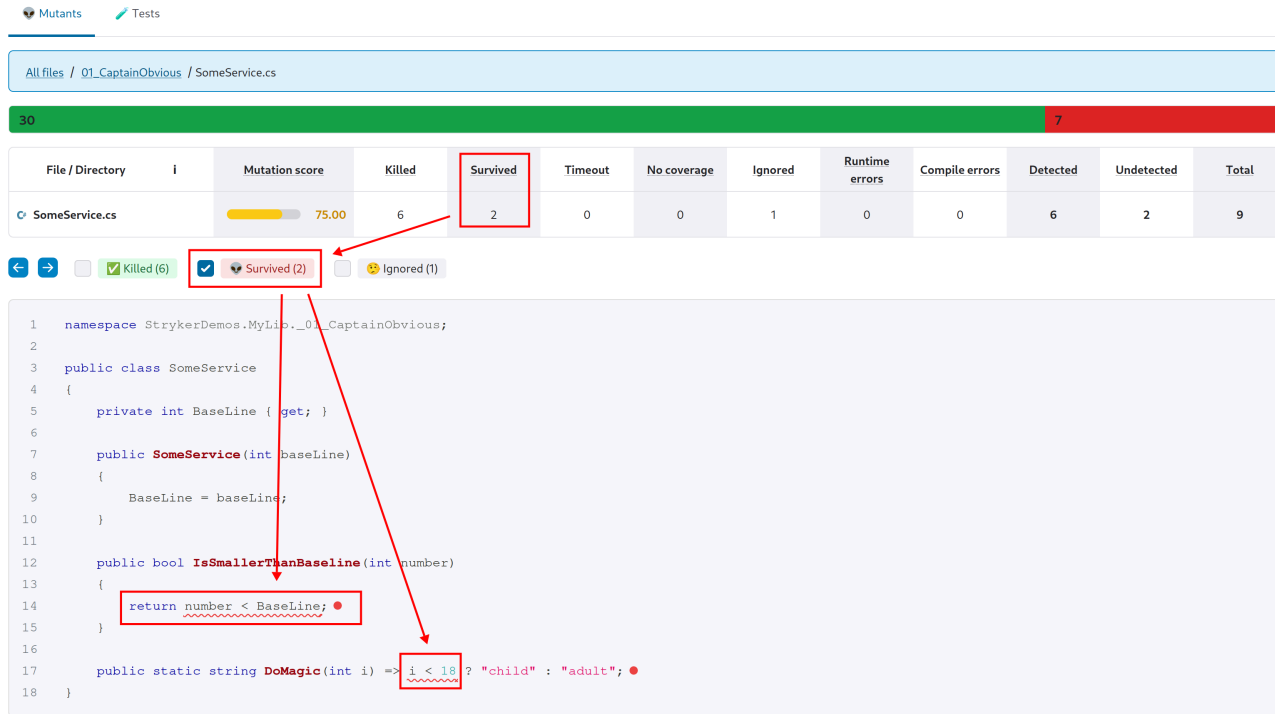
7

File / Directory	i	Mutation score	Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
📁 All files		81.08	30	7	0	0	10	0	1	30	7	48
🔗 00_FizzBuzz/FizzBuzzer.cs		100.00	14	0	0	0	4	0	0	14	0	18
🔗 01_CaptainObvious/SomeService.cs		75.00	6	2	0	0	1	0	0	6	2	9
🔗 02_OrderProcessing/OrderProcessor.cs		72.73	8	3	0	0	3	0	0	8	3	14
🔗 03_Palindrome/PalindromeChecker.cs		50.00	2	2	0	0	2	0	1	2	2	7



# Reports: HTML (Details)

## SomeService.cs Stryker.NET Report





## Other Reporters

- Json (basis for HTML)
- Progress
- Cleartext
- Cleartext tree
- Dots (for CI)
- Markdown
- Dashboard

# My Stryker Dashboard



Mutation score unknown



# Fine-Tuning

🔧 Stryker provides many bells & whistles for fine-tuning using either CLI or config file.

Some examples:

- ``mutate`` : Globbing patterns for including/excluding
- ``test-case-filter`` : filter selective subset(s) of tests
- ``mutation-level`` : high level categories ( ``Basic`` , ``Standard`` , ``Advanced`` , ``Complete`` )
- ``coverage-analysis`` : short circuit logic vs "everything in isolation"

Also nice: use git as baseline, only test things that have changed recently

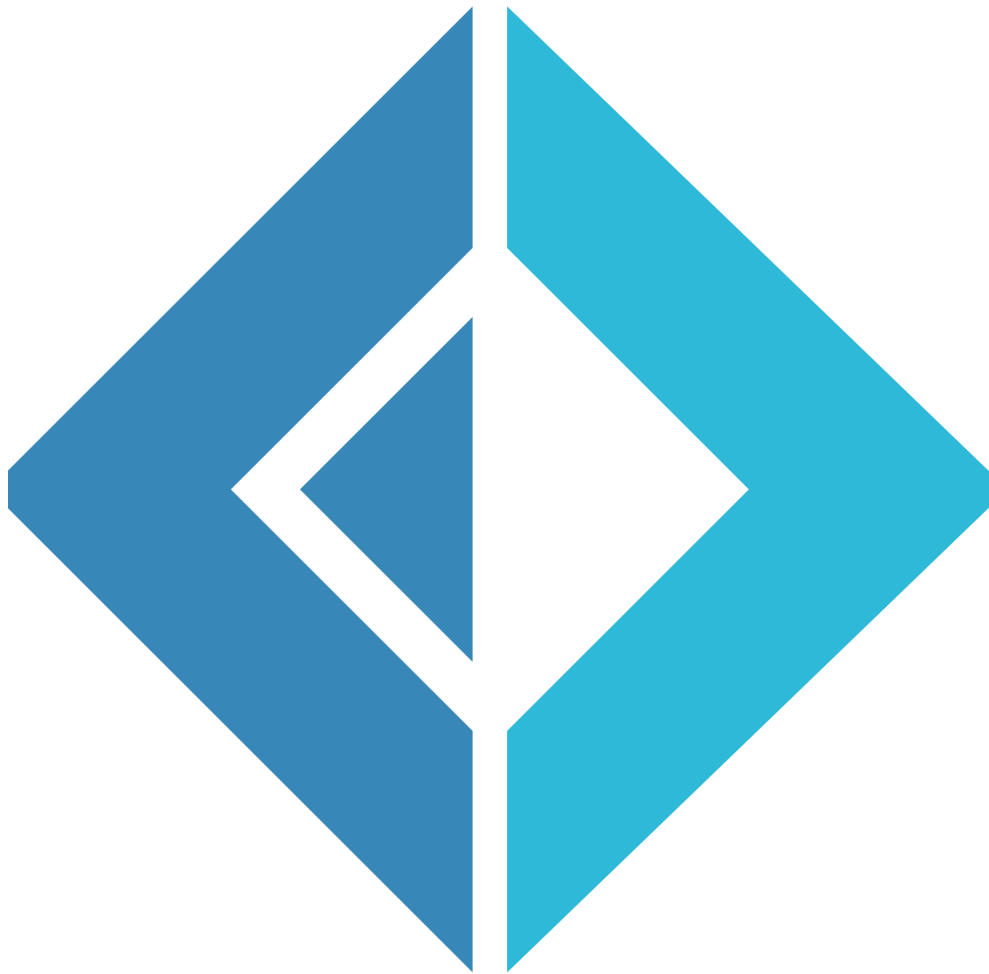
- ``since`` : git "committish" (i.e. commit hash, tag, etc)
- ``with-baseline`` (experimental): similar to ``since`` , but uses previous reports





## What about F#?









- The team noticed they had to rearchitect the framework (.NET is not only C#)
- This is a good thing!
- Strategy is clearly communicated!
- 🎉





# Mutation Testing: Available in many languages

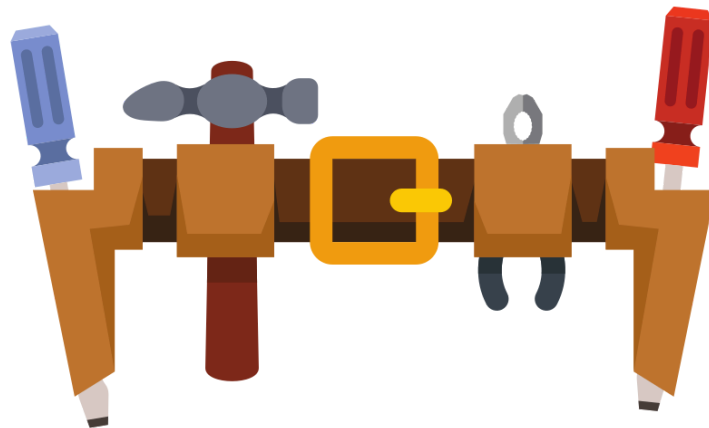
Overview: <https://github.com/theofidry/awesome-mutation-testing>

-  JavaScript: <https://stryker-mutator.io/docs/stryker-js/>
-  Scala: <https://stryker-mutator.io/docs/stryker4s/>
-  Java: <https://pitest.org/>
-  Python: <https://mutatest.readthedocs.io>
-  C/C++: <https://github.com/mull-project/mull>
-  Rust: <https://mutants.rs/>
-  Go: <https://github.com/zimmski/go-mutesting>
-  Haskell: <https://hackage.haskell.org/package/MuCheck>
- etc. (search for "your-programming-language mutation test")








# Mutation Testing: Summary

- 🕶️ none-invasive: no code changes required!
- 🔍 great for discovering important corner cases
- 😞 requires a lot of resources: use wisely
- 🎉 great addition to our "Testing Toolbelt"
  - Test-Driven Development (TDD)
  - Approval Testing
  - Property-Based Testing (PBT)





# Thank You!

-  [patrick.drechsler@mathema.de](mailto:patrick.drechsler@mathema.de)
-  <https://github.com/draptik>
-  Blog: <https://draptik.github.io>
-  [@drechsler@floss.social](https://twitter.com/drechsler@floss.social)
-  <https://www.linkedin.com/in/patrick-drechsler-draptik/>

Slides 

- QR Code or
- <https://draptik.github.io/2024-07-dwx24-mutation-testing>
- sample code: <https://github.com/draptik/2024-mutation-testing>

Image sources: [pixabay.com](https://pixabay.com) and [perchance.org](https://perchance.org)

