Mutation Testing

Rage against the machine.

Agenda

1. Introducing Mutation Testing
2. Manual Mutation Testing
3. Introduction to Stryker
4. What does Stryker do?
5. Mutation Types
6. Applying Mutations to Code
7. Live Demo

First, some (possibly) controversial opinions

1. Writing tests is easy

1. Writing tests is easy

2. Knowing what to test is hard

1. Writing tests is easy

2. Knowing what to test is hard

3. Preventing regressions is even harder

1. Writing tests is easy
2. Knowing what to test is hard
3. Preventing regressions is even harder
4. Code coverage is a bad metric for test quality

Introducing Mutation Testing

- What if we could test our tests?

Introducing Mutation Testing

- What if we could test our tests?
- Mutation testing evaluates the quality of your test suite

Introducing Mutation Testing

- What if we could test our tests?
- Mutation testing evaluates the quality of your test suite
- Try to break your code and see if your tests catch it
  - If your tests fail, they cover the mutation
  - If your tests pass, add more tests

Manual Mutation Testing

```
1  public bool IsOldEnoughToDrink(int age)
2  {
3      if (age ≥ 18)
4          return true;
5
6      return false;
7  }
```

- ✅ `age = 17 → false`
- ✅ `age = 19 → true`

Our code 100% statement coverage

Manual Mutation Testing

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age ≥ 18)
4         return true;
5
6     return false;
7 }
```

- ✅ `age = 17 → false`
- ✅ `age = 19 → true`

Did we cover all possible conditions?

Manual Mutation Testing

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age > 18)
4         return true;
5
6     return false;
7 }
```

`age = 17 → false`

`age = 19 → true`

We change `≥` to `>`

Manual Mutation Testing

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age > 18)
4         return true;
5
6     return false;
7 }
```

✅ `age = 17 → false`

✅ `age = 19 → true`

⚠️ No failures → our tests are not complete

Manual Mutation Testing

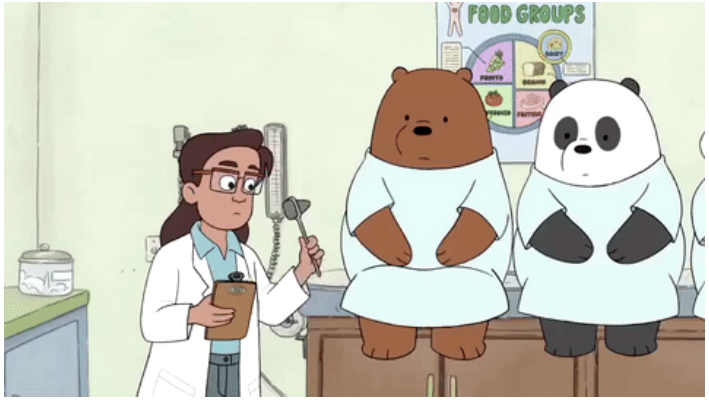
```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age > 18)
4         return true;
5
6     return false;
7 }
```

- ✅ `age = 17 → false`
- ✅ `age = 19 → true`
- ❌ `age = 18 → true`

We add an extra test to catch the mutation

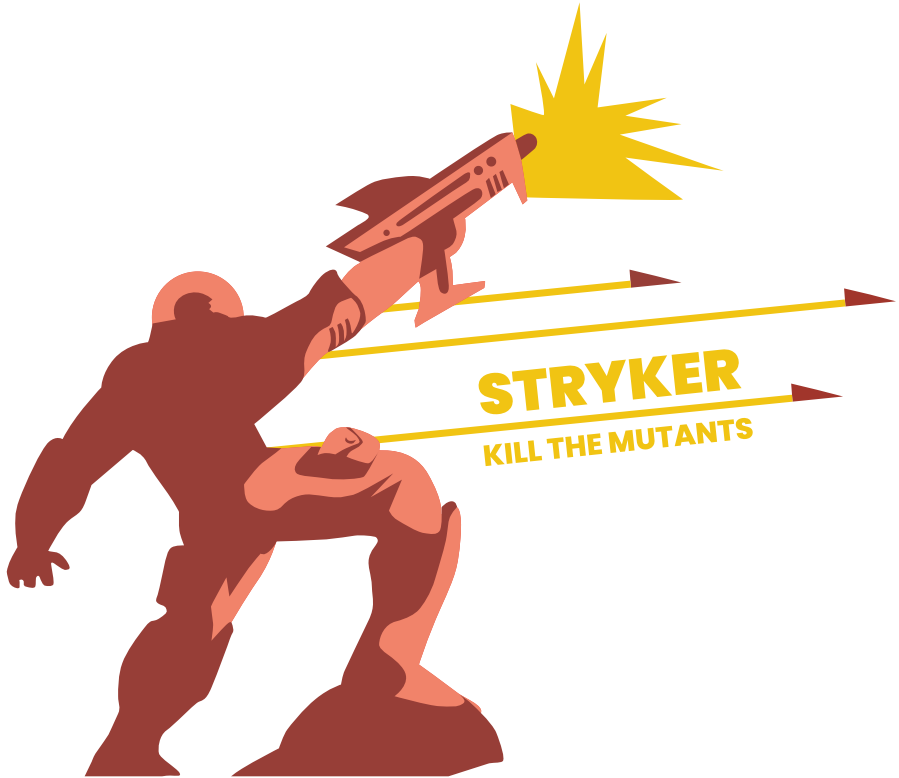
Manual Mutation Testing is tedious

We can do better!



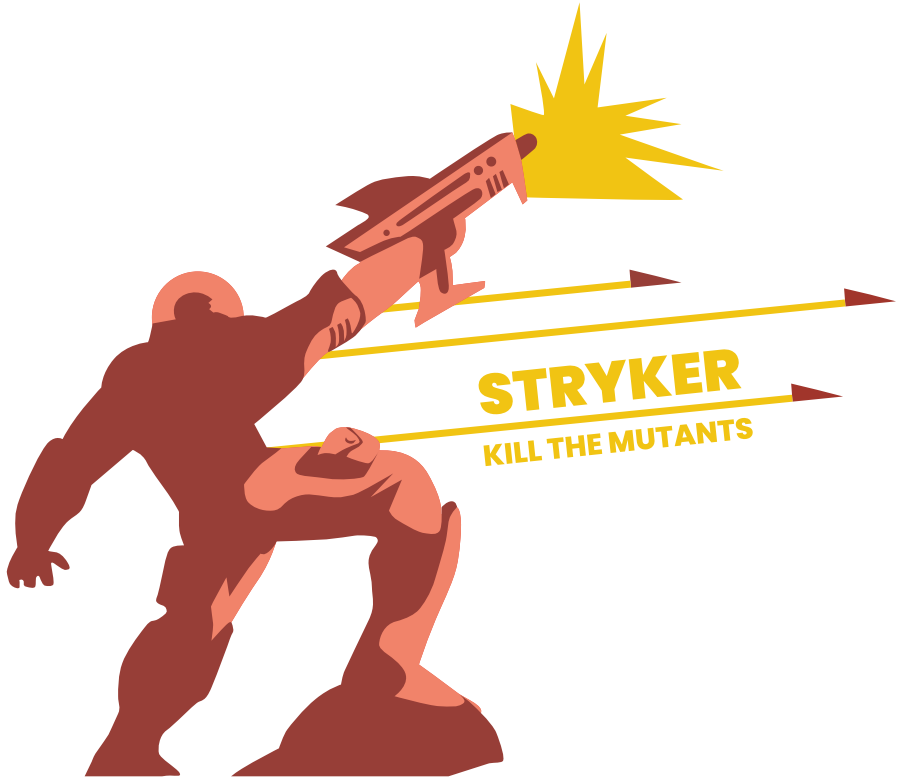
Introduction to Stryker

- Open-source mutation testing tool



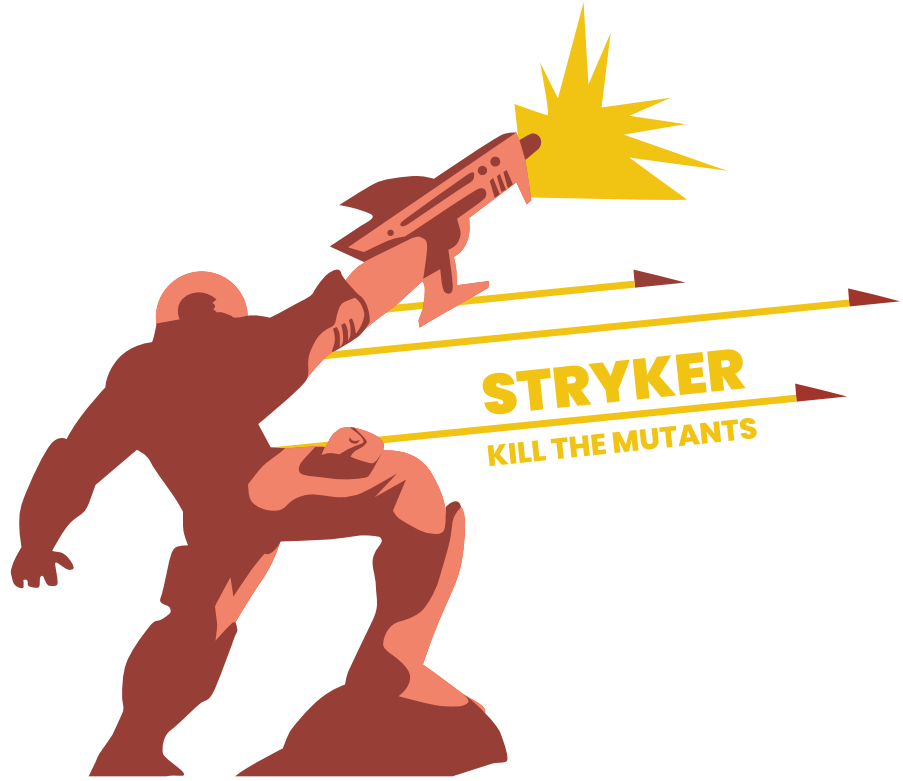
Introduction to Stryker

- Open-source mutation testing tool
- Runs using your existing test suite



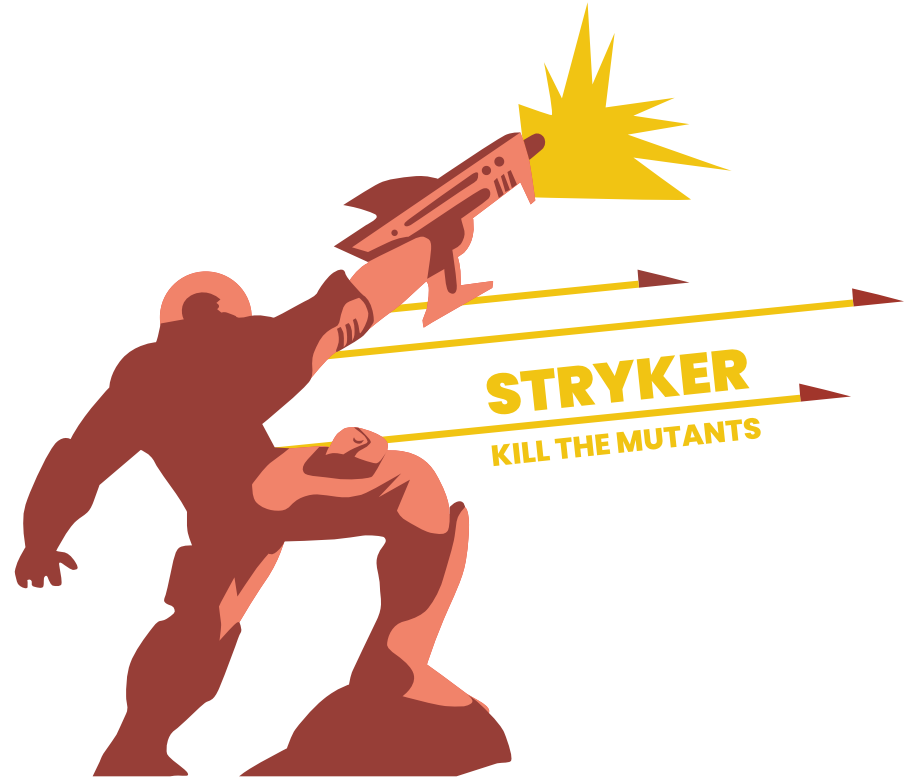
Introduction to Stryker

- Open-source mutation testing tool
- Runs using your existing test suite
- Supports multiple languages
  - JavaScript / TypeScript
  - C# / .NET
  - Scala



Introduction to Stryker

- Open-source mutation testing tool
- Runs using your existing test suite
- Supports multiple languages
  - JavaScript / TypeScript
  - C# / .NET
  - Scala
- Can integrate into CI/CD pipelines
  - Set thresholds for when to fail the build



What does Stryker do?

1. Reads your code
2. Decides possible mutations

ShippingCostCalculator.cs Stryker.NET Report

Mutants | Tests

All files / ShippingCostCalculator.cs

6 | 3

File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
ShippingCostCalculator.cs	66.67	66.67	6	3	0	0	3	0	0	6	3	12

Killed (6) | Survived (3) | Ignored (3)

```
1   namespace MutationTesting.Domain;
2
3   public class ShippingCostCalculator
4   {
5       private static decimal MaxFreeWeightInKg = 5.00m;
6
7       const decimal BaseFee = 1.00M;
8       const decimal HeavyWeightFee = 1.50m;
9       const decimal ExpressFee = 7.50m;
10
11      private readonly Dictionary<string, decimal> _cache = new();
12
13      public decimal CalculateShippingCost(decimal weightInKg, bool expressShipping = false)
14      {
15          var cacheKey = $"{weightInKg}:{expressShipping}";
16
```

What does Stryker do?

1. Reads your code
2. Decides possible mutations
3. Applies mutations
4. Runs your tests

ShippingCostCalculator.cs Stryker.NET Report

Mutants | Tests

All files / ShippingCostCalculator.cs

6 | 3

File / Directory	Mutation Score Of total	Mutation Score Of covered	Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
ShippingCostCalculator.cs	66.67	66.67	6	3	0	0	3	0	0	6	3	12

Killed (6) | Survived (3) | Ignored (3)

```
1   namespace MutationTesting.Domain;
2
3   public class ShippingCostCalculator
4   {
5       private static decimal MaxFreeWeightInKg = 5.00m;
6
7       const decimal BaseFee = 1.00M;
8       const decimal HeavyWeightFee = 1.50m;
9       const decimal ExpressFee = 7.50m;
10
11      private readonly Dictionary<string, decimal> _cache = new();
12
13      public decimal CalculateShippingCost(decimal weightInKg, bool expressShipping = false)
14      {
15          var cacheKey = $"{weightInKg}:{expressShipping}";
16
```

What does Stryker do?

1. Reads your code
2. Decides possible mutations
3. Applies mutations
4. Runs your tests
5. Checks if tests fail
6. Reports results

ShippingCostCalculator.cs Stryker.NET Report

Mutants | Tests

All files / ShippingCostCalculator.cs

6 | 3

File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
ShippingCostCalculator.cs	66.67	66.67	6	3	0	0	3	0	0	6	3	12

Killed (6) | Survived (3) | Ignored (3)

```
1   namespace MutationTesting.Domain;
2
3   public class ShippingCostCalculator
4   {
5       private static decimal MaxFreeWeightInKg = 5.00m;
6
7       const decimal BaseFee = 1.00M;
8       const decimal HeavyWeightFee = 1.50m;
9       const decimal ExpressFee = 7.50m;
10
11      private readonly Dictionary<string, decimal> _cache = new();
12
13      public decimal CalculateShippingCost(decimal weightInKg, bool expressShipping = false)
14      {
15          var cacheKey = $"{weightInKg}:{expressShipping}";
16
```

Mutation Types

Mutation Type	Description

Mutation Types

Mutation Type	Description
Equality Operator Replacement	Swap `==`, `≠`, `>`, `<`, `⩾`, `⩽`
Arithmetic Operator Replacement	Swap `+`, `-`, `*`, `/`, `%`

Mutation Types

Mutation Type	Description
Equality Operator Replacement	Swap `==`, `≠`, `>`, `<`, `⩾`, `⩽`
Arithmetic Operator Replacement	Swap `+`, `-`, `*`, `/`, `%`
Logical Operator Replacement	Swap `!`, `&&`, `||`, `and`, `is`, `is not`, `or`
Boolean Literal Replacement	Swap `true` & `false`, replace `cond` with `!cond`

Mutation Types

Mutation Type	Description
Equality Operator Replacement	Swap `==`, `!=`, `>`, `<`, `>=`, `<=`
Arithmetic Operator Replacement	Swap `+`, `-`, `*`, `/`, `%`
Logical Operator Replacement	Swap `!`, `&&`, `||`, `and`, `is`, `is not`, `or`
Boolean Literal Replacement	Swap `true` & `false`, replace `cond` with `!cond`
Assignment Operator Replacement	Swap `=`, `+=`, `-=`, `*=`, `/=`, `%=`
Initialization Mutators	Replace initializers like `[1, 2, 3]` with `[]`

Mutation Types

Mutation Type	Description
Equality Operator Replacement	Swap `==`, `!=`, `>`, `<`, `>=`, `<=`
Arithmetic Operator Replacement	Swap `+`, `-`, `*`, `/`, `%`
Logical Operator Replacement	Swap `!`, `&&`, `||`, `and`, `is`, `is not`, `or`
Boolean Literal Replacement	Swap `true` & `false`, replace `cond` with `!cond`
Assignment Operator Replacement	Swap `=`, `+=`, `-=`, `*=`, `/=`, `%=`
Initialization Mutators	Replace initializers like `[1, 2, 3]` with `[]`
Removal Mutators	Remove statements and blocks (`return`, `break`, `throw`)

Applying Mutations to Code

```
1  public bool IsOldEnoughToDrink(int age)
2  {
3      if (age ≥ 18)
4          return true;
5
6      return false;
7  }
```

The original code

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (!(age ≥ 18))
4         return true;
5
6     return false;
7 }
```

Negate expression mutation

`condition` to `!condition`

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age > 18)
4         return true;
5
6     return false;
7 }
```

Equality mutation

`⩾` to `>`

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age < 18)
4         return true;
5
6     return false;
7 }
```

Equality mutation

`⩾` to `<`

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age ≥ 18)
4         return false;
5
6     return false;
7 }
```

Boolean mutation

`true` to `false`

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age ≥ 18)
4         return true;
5
6     return true;
7 }
```

Boolean mutation

`false` to `true`

Applying Mutations to Code

```
1 public bool IsOldEnoughToDrink(int age)
2 {
3     if (age ≥ 18)
4         /* Do nothing */
5
6     return false;
7 }
```

Block removal mutation

Remove the `if` block

Applying Mutations to Code

This "simple" `if` statement results in 6 possible mutations

Live Demo

You can try this at home!

Clone the example repo (or just download the code)

```
git clone https://github.com/wvanlit/mutation-testing-with-stryker.git
```

TypeScript

```
# Install dependencies
npm install

# Run mutation testing
npm run test:mutation
```

C#

```
# Install dependencies
dotnet restore

# Run mutation testing
dotnet stryker
```

Conclusion

- Mutation testing is a powerful tool
- Helps you write better tests
- Can be integrated into your CI/CD pipeline
- Stryker is a great tool to get started

Conclusion

- Mutation testing is a powerful tool
- Helps you write better tests
- Can be integrated into your CI/CD pipeline
- Stryker is a great tool to get started

But remember…

- Mutation testing is not a silver bullet
- It’s not a replacement for good testing practices

Questions?

Ask!