

▲ Download the repo for labs and slides



[https://github.com/
OssamaSijbesma/stryker-labs](https://github.com/OssamaSijbesma/stryker-labs)





Kill all mutants with Stryker

Mutation testing workshop

15-11-2022

▲ Ossama Sijbesma


**Software engineer
& AI-community lead**

Ossama.Sijbesma@infosupport.com





▲ Agenda

1. Introduction
 2. Mutation testing
 3. Stryker mutator
 4. Hands-on lab
- 



Info Support



Consultancy

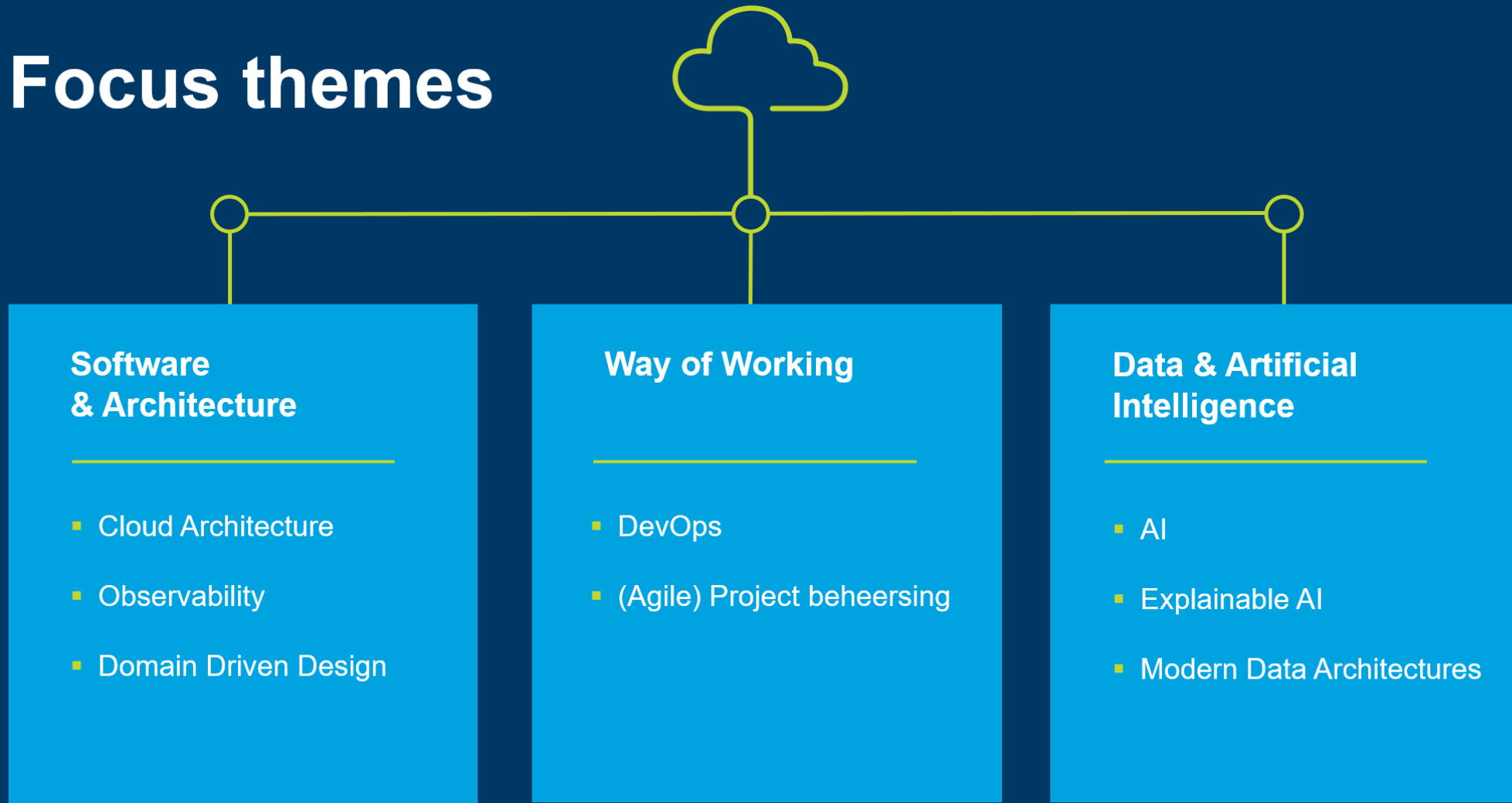


Multiple sectors



~ 500 colleagues

Focus themes



Work & Internships

- Dirk Spanbroek

<https://www.linkedin.com/in/dirkspanbroek/>

- Or visit our website

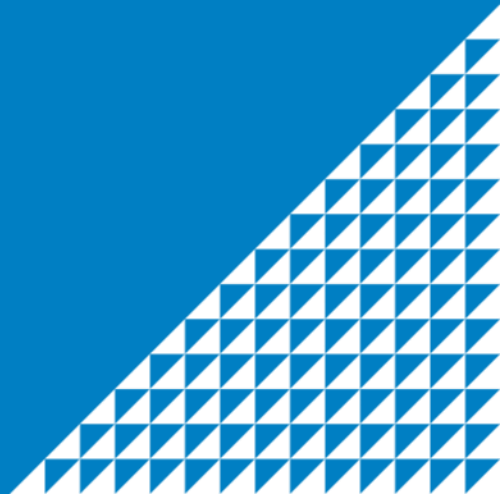
<https://carriere.infosupport.com/>

(I would appreciate it if you mention me!)





Unit Testing



▲ When are my unit tests good?

 Testing patterns

 All tests are green

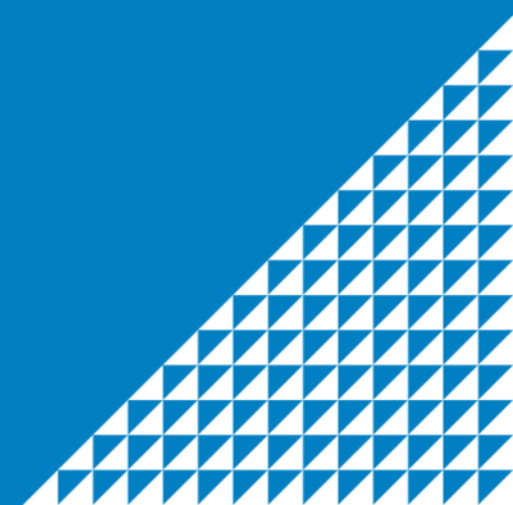
 > 80% code coverage

 I have great tests

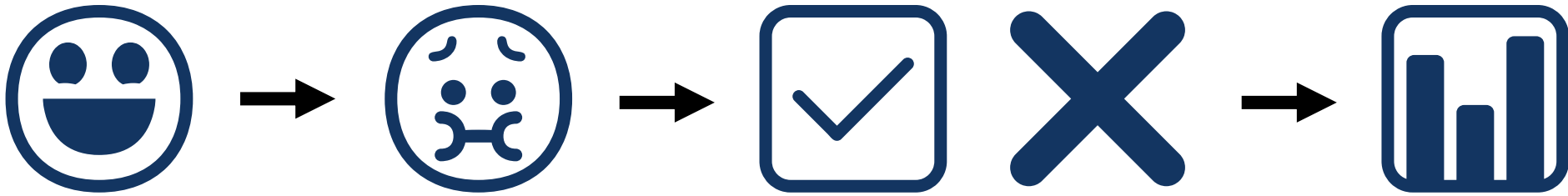




Mutation testing



▲ Mutation testing process



1. Source code
2. Introduce mutant
3. Test (killed / survived)
4. Report

▲ Mutating code

```
// Production Code
public static bool IsAllowedToBuyAlcohol(Customer customer) {
    return customer.Age >= 18;
    // ☑ Test succeeds
}

// Test
[TestMethod]
public void CustomerIsOlderThan18_ReturnTrue() {
    Customer customer = new Customer("Professor X", 96);
    Assert.IsTrue(Store.IsAllowedToBuyAlcohol(customer));
}
```

▲ Mutating code

```
// Production Code
public static bool IsAllowedToBuyAlcohol(Customer customer) {
    return customer.Age < 18;
    // ✕ Test fails, mutant KILLED
}

// Test
[TestMethod]
public void CustomerIsOlderThan18_ReturnTrue() {
    Customer customer = new Customer("Professor X", 96);
    Assert.IsTrue(Store.IsAllowedToBuyAlcohol(customer));
}
```


▲ Mutating code

```
// Production Code
public static bool IsAllowedToBuyAlcohol(Customer customer) {
    return customer.Age > 18;
    // ☑ Test succeeds, mutant SURVIVED
}

// Test
[TestMethod]
public void CustomerIsOlderThan18_ReturnTrue() {
    Customer customer = new Customer("Professor X", 96);
    Assert.IsTrue(Store.IsAllowedToBuyAlcohol(customer));
}
```

▲ Mutating code

```
// Production Code
public static bool IsAllowedToBuyAlcohol(Customer customer) {
    return true;
    // ☑ Test succeeds, mutant SURVIVED
}

// Test
[TestMethod]
public void CustomerIsOlderThan18_ReturnTrue() {
    Customer customer = new Customer("Professor X", 96);
    Assert.IsTrue(Store.IsAllowedToBuyAlcohol(customer));
}
```

Common mutations

Original	Mutated
<code>a + b</code>	<code>a - b</code>
<code>a / b</code>	<code>a * b</code>
<code>a < b</code>	<code>a > b</code>
<code>a == b</code>	<code>a != b</code>
<code>a && b</code>	<code>a b</code>
<code>string drink = "Cola"</code>	<code>string drink = ""</code>
<code>int[] list = {1, 2, 3, 4}</code>	<code>int[] list = {}</code>
<code>if (a > b) { ... }</code>	<code>if (true) { ... }</code>
<code>public void fn() { ... }</code>	<code>public void fn() { /* EMPTY */ }</code>

▶ Mutant states

- ☑ Killed
- 👁 Survived
- 👁 No coverage
- ⌚ Timeout
- 💥 Runtime error
- 💥 Compile error
- 😐 Ignored

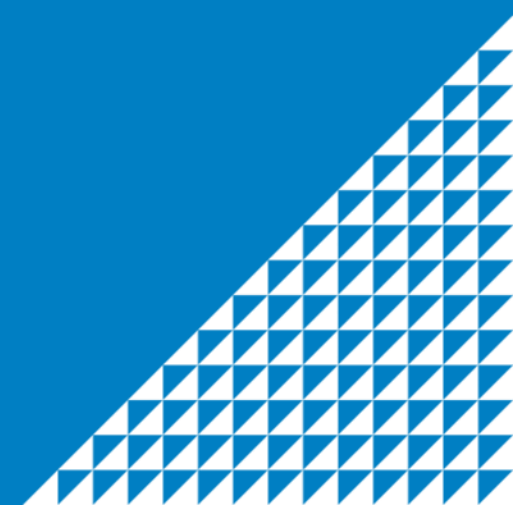


▲ Mutant metrics

- Detected
killed + timeout
- Undetected
survived + no coverage
- Covered
detected + survived
- Valid
detected + undetected
- Invalid
Runtime Error + Compile Error
- Mutation score
 $\text{detected} / \text{valid} * 100$
- Mutation score based on covered code
 $\text{detected} / \text{covered} * 100$



Question: what different conclusions can we derive from the 2 scores?



▲ Frameworks

Language	Framework
JavaScript & TypeScript	StrykerJS
Scala	Stryker4s
C#	Stryker.NET
Java	PITest
PHP	InfectionPHP
Ruby	Mutant
Python	Cosmic Ray
C/C++	Mull

▲ Disadvantages

-  Slower
-  Configuration
-  Project support



Improving performance

$$T_{\text{total}} \neq T_{\text{test}} * N$$

- ▶ Do faster
- ▶ Do fewer
- 🤔 Do smarter

▶ Do faster

“Reduce execution time by using novel algorithms, tool improvements, or special-purpose hardware”

~ 30 papers

▶ Do fewer

“The objective is to reduce the number of mutants that will be executed, preferably without reducing effectiveness”

~ 65 papers



◀ 🤔 Do smarter

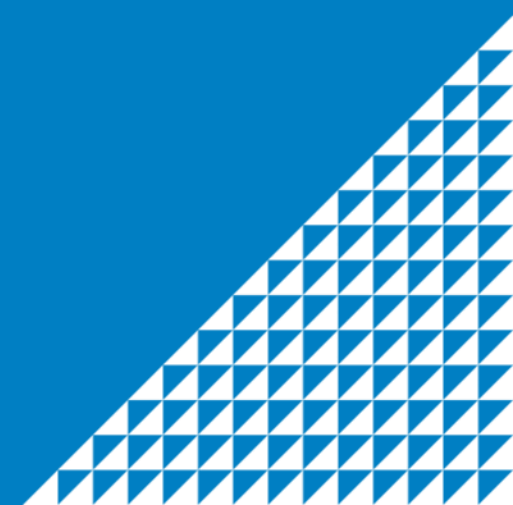
“The objective is either to find smaller test sets that are still as effective at killing mutants, or to identify groups of similar mutants to reduce the number of test runs”

~ 15 papers



Mutating strategies

How to place the mutations into the code

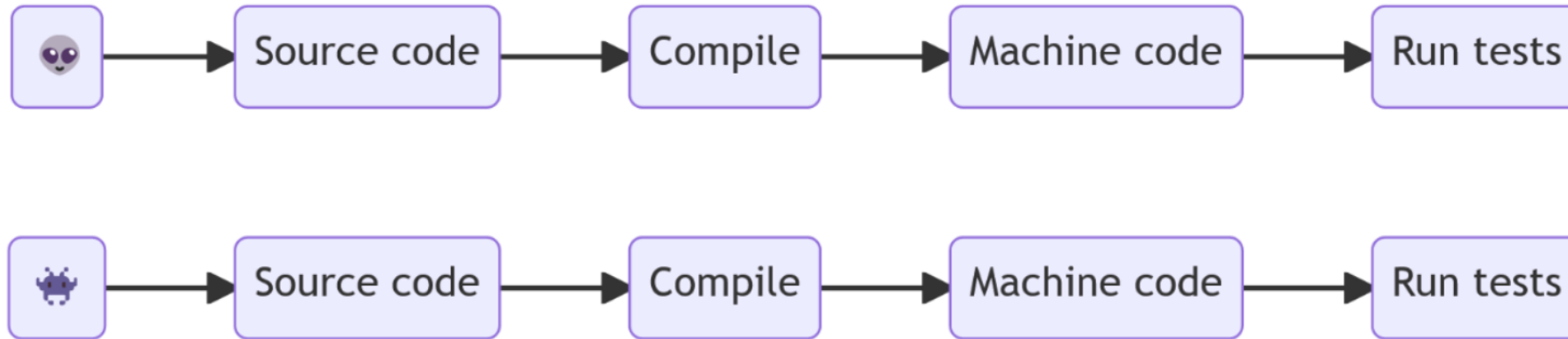


▲ 2 obvious candidates

1. Source code mutation
2. Byte code mutation



▲ Source code mutation

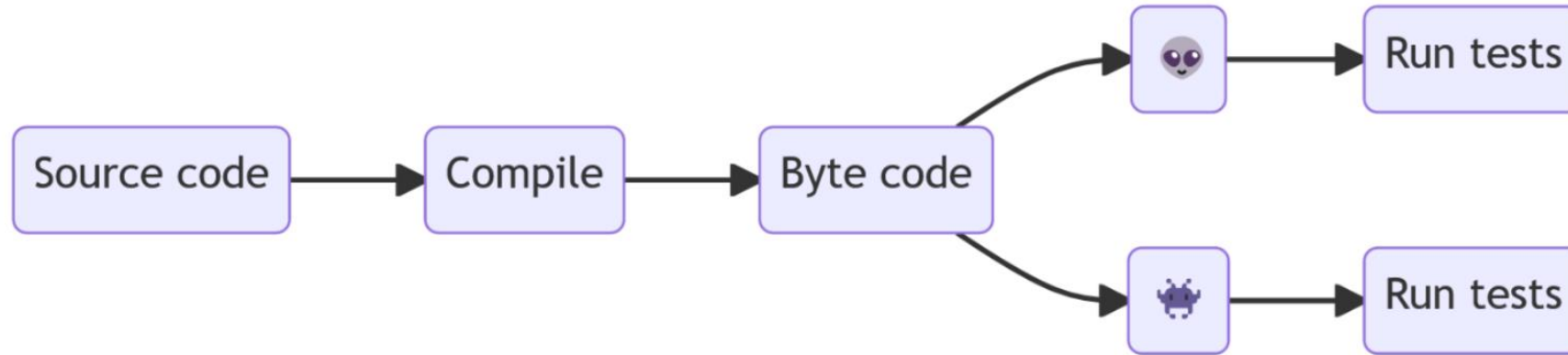


- Generate mutants based on source code

- ☒ Precise
- ☒ Easy
- ☒ Slow



Byte code mutation



- Generate mutants based on compiled code

- ☑ Fast...ish
- ✗ False positives
- ✗ Complicated



▲ Mutant schemata

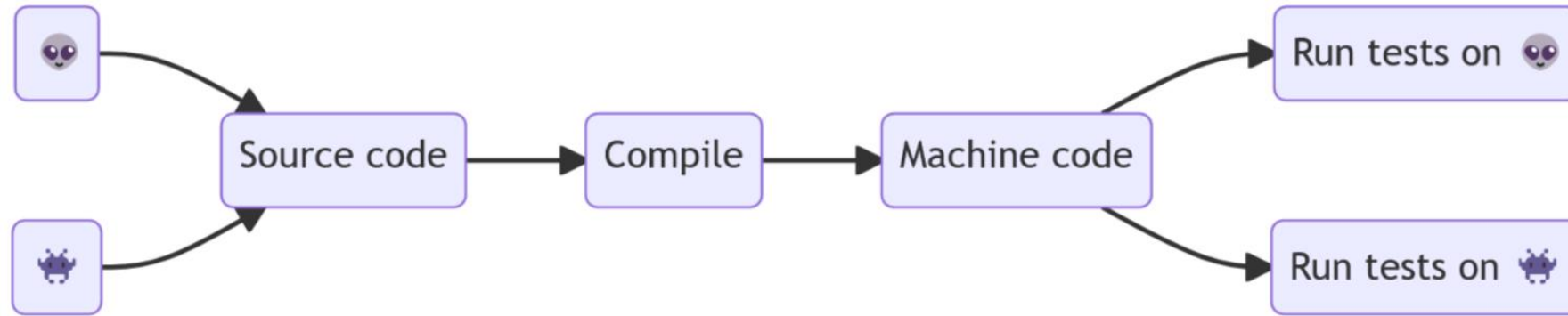
Mutation Analysis Using Mutant Schemata

Roland H. Untch
Department of Computer Science
Clemson University
Clemson, SC 29634-1906
untch@cs.clemson.edu

A. Jefferson Offutt
ISSE Department
George Mason University
Fairfax, VA 22030-4444
ofut@isse.gmu.edu

Mary Jean Harrold
Department of Computer Science
Clemson University
Clemson, SC 29634-1906
harrold@cs.clemson.edu

▲ Mutant schemata process



- Generate mutants based on source code, but compile once

- ☒ Precise
- ☒ Fast
- ☐ Complicated (but manageable)



▲ Mutation testing conclusion

- Testing the tests
- Mutation score: how many mutants were detected
- Framework support is broad
- A lot of research on performance
- There are 3 mutation strategies
 - Mutant schemata is generally the best approach







Stryker

Mutation testing

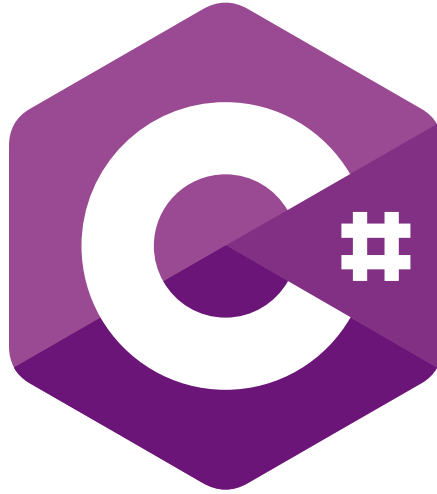
“

For someone who hates mutants, you certainly keep some strange company.
— Professor X

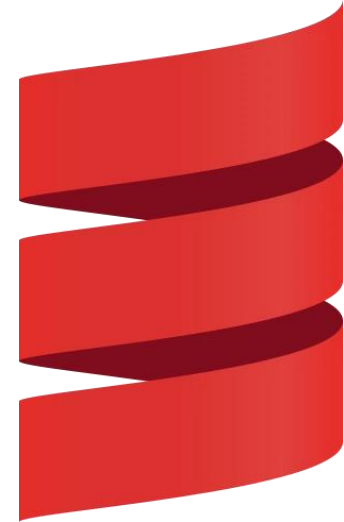
Oh, they serve their purpose. As long as they can be controlled.
— William Stryker



JavaScript
and friends

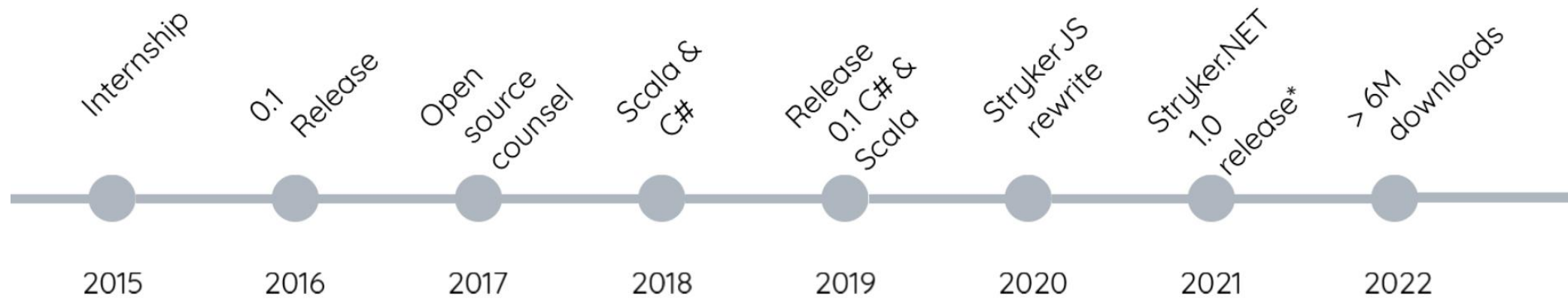


C#



Scala

▲ Origin story



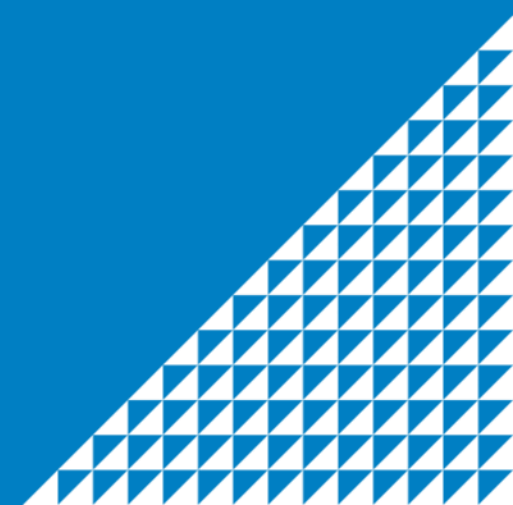


▲ Some highlights

- StrykerJS
 - More than 5m total downloads
 - 60.654 downloads a week
- Stryker.NET and Stryker4s
 - More than 0.5m total downloads each
- Shared projects
 - `mutation-testing-elements`: HTML report for mutation testing
 - `stryker-dashboard`: Dashboard for mutation testing reports
 - `weapon-regex`: Regex mutations for Scala & JavaScript



Stryker.NET Mutators



▲ Arithmetic Operators

Original	Mutated
+	-
-	+
*	/
/	*
%	*

Equality Operators

Original	Mutated
>	<
>	>=
>=	<
>=	>
<	>
<	<=
<=	>
<=	<
==	!=
!=	==

▲ Logical Operators

Original	Mutated
&&	
	&&
^	==

Boolean Literals

Original	Mutated
true	false
false	true
!person.IsAdult()	person.IsAdult()
if(person.IsAdult())	if(!person.IsAdult())
while(person.IsAdult())	while(!person.IsAdult())

Assignment Statements

Original	Mutated
<code>+=</code>	<code>-=</code>
<code>-=</code>	<code>+=</code>
<code>*=</code>	<code>/=</code>
<code>/=</code>	<code>*=</code>
<code>%=</code>	<code>*=</code>
<code><<=</code>	<code>>>=</code>
<code>>>=</code>	<code><<=</code>
<code>&=</code>	<code> =</code>
<code>&=</code>	<code>^=</code>
<code> =</code>	<code>&=</code>
<code> =</code>	<code>^=</code>
<code>^=</code>	<code> =</code>
<code>^=</code>	<code>&=</code>

Initialization

Original	Mutated
<code>new int[] { 1, 2 };</code>	<code>new int[] { };</code>
<code>int[] numbers = { 1, 2 };</code>	<code>int[] numbers = { };</code>
<code>new List<int> { 1, 2 };</code>	<code>new List<int> { };</code>
<code>new Collection<int> { 1, 2 };</code>	<code>new Collection<int> { };</code>
<code>new Dictionary<int, int> { { 1, 1 } };</code>	<code>new Dictionary<int, int> { };</code>
<code>new SomeClass { Foo = "Bar" };</code>	<code>new SomeClass { };</code>

Removal mutators

Original	Mutated
<code>void Function() { Age++; }</code>	<code>void Function() {}</code> (block emptied)
<code>int Function() { Age++; return Age; }</code>	<code>void Function() { return default; }</code> (block emptied)
<code>return;</code>	removed
<code>return value;</code>	removed
<code>break;</code>	removed
<code>continue;</code>	removed
<code>goto;</code>	removed
<code>throw;</code>	removed
<code>throw exception;</code>	removed
<code>yield return value;</code>	removed
<code>yield break;</code>	removed
<code>MyMethodCall();</code>	removed

Unary Operators

Original	Mutated
-variable	+variable
+variable	-variable
~variable	variable

Update Operators

Original	Mutated
Original	Mutated
variable++	variable--
variable--	variable++
++variable	--variable

▲ Checked Statements

Original	Mutated
checked(2 + 4)	2 + 4

▲ Linq Methods

Original	Mutated
SingleOrDefault()	Single()
Single()	SingleOrDefault()
FirstOrDefault()	First()
First()	FirstOrDefault()
Last()	First()
All()	Any()
Any()	All()
Skip()	Take()
Take()	Skip()
SkipWhile()	TakeWhile()
TakeWhile()	SkipWhile()
Min()	Max()

Original	Mutated
Max()	Min()
Sum()	Max()
Count()	Sum()
Average()	Min()
OrderBy()	OrderByDescending()
OrderByDescending()	OrderBy()
ThenBy()	ThenByDescending()
ThenByDescending()	ThenBy()
Reverse()	AsEnumerable()
AsEnumerable()	Reverse()
Union()	Intersect()
Intersect()	Union()

▲ String Literals and Constants

Original	Mutated
"foo"	""
"""	"Stryker was here!"
`\${foo {bar}}`	`\${}`
@`foo`	@`""`
string.Empty	"Stryker was here!"

▲ Bitwise Operators

Original	Mutated
<<	>>
>>	<<
&	
	&
a^b	$\sim(a^b)$



▲ Regular Expressions

Types
Common tokens
Anchors
Quantifiers
Group constructs



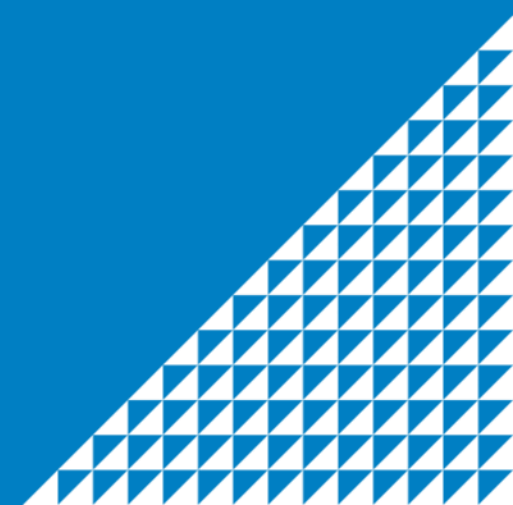
Questions?

Ossama.Sijbesma@infosupport.com





Stryker.NET Internals



▲ Choosing the mutation strategy

🙄 Mutant schemata

However, this is not possible...

- Mutating constant values
- Mutating method names
- Mutating access modifiers





How do we actually add mutants?

We currently know:

- What mutation testing is, testing the tests
- Which strategies exist (mutant schemata best)



▲ Roslyn API

- 🔍 Analyzers (code smells, compile errors, etc.)
- 🔧 Code fixes (find and correct code)

Use cases:

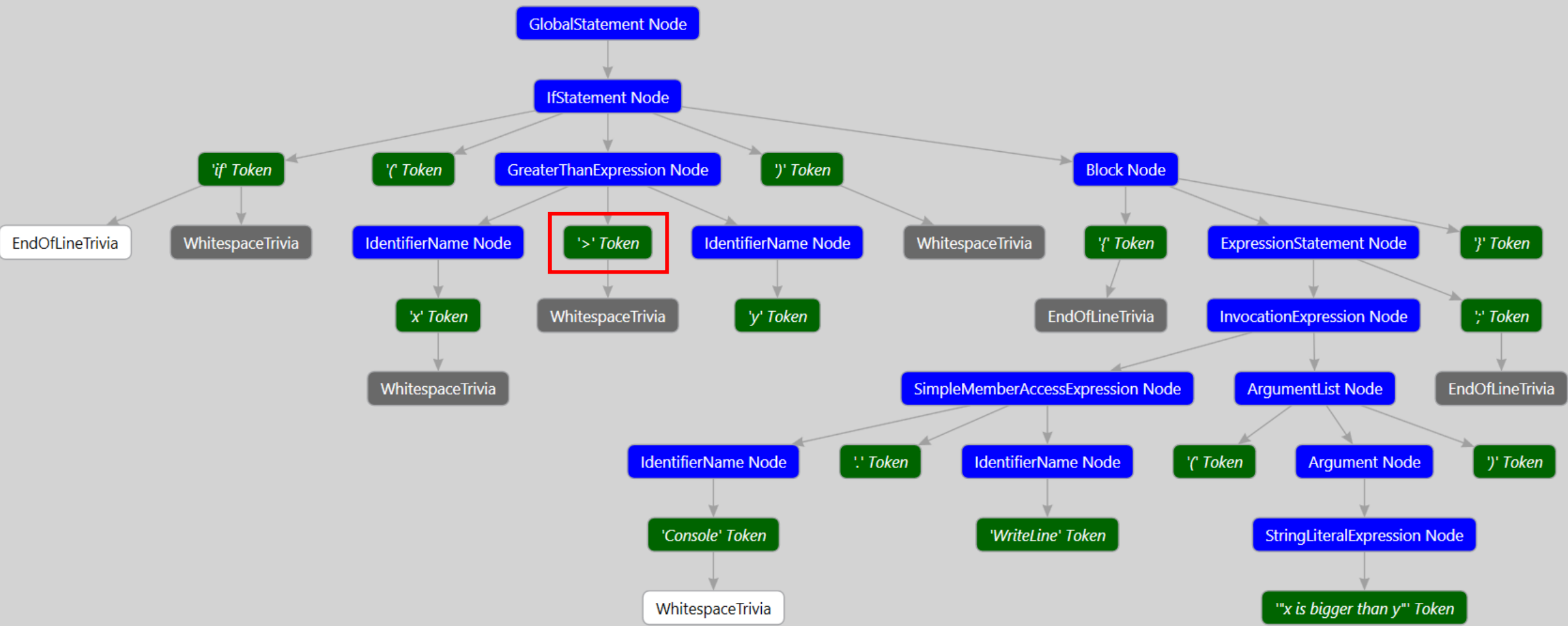
- ☑ Code completion (IntelliSense)
- ❓ Inline parameter & type hints
- ↻ Code refactoring's
- 🔬 Mutation testing



▲ Syntax Trees

- Data structure used by the compiler
- 4 Primary building blocks:
 1. 🌳 Syntax tree (an instance of which represents an entire parse tree)
 2. 🌱 Syntax node (declarations, statements, clauses, and expressions)
 3. 📄 Syntax Token (individual keyword, identifier, operator, or punctuation)
 4. 🌟 Syntax Trivia (white space between tokens, preprocessing directives, and comments.)

```
[-] if (x > y) {  
    |     Console.WriteLine("x is bigger than y");  
    | }  
    |
```





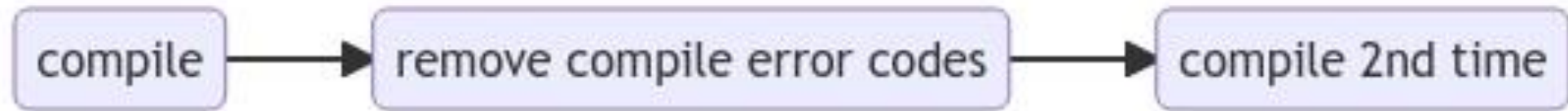
▲ Compile errors

```
if (Environment.GetEnvironmentVariable("ActiveMutation") == "1")
{
    return "hello " - "world"; // mutated code
}
else
{
    return "hello " + "world"; // original code
}
```

✗ Not all mutations can be compiled



▲ Rollback



- 🔄 Rollbacking all mutations that result in compile errors
- 🔄 Repeat process until we compile the code
- 😬 Mutant gets status build error

▲ Scope

```
if (Environment.GetEnvironmentVariable("ActiveMutation") == "1")
{
    int i = 0; // mutated code
}
else
{
    int i = 99; // original code
}
return i;
```

✗ Cannot be compiled because the scope of the variables can change when places inside an if statement

▲ Using conditional statements

🔄 Rollback is not possible

🔧 Use conditional statements instead of if statements

```
int i = Environment.GetEnvironmentVariable("ActiveMutation") == "1" ? 0 : 99;  
return i;
```

▲ Constant values

```
[-] public enum Numbers
{
    One = 1,
    Two = (One + 1)
}
```

```
[-] public enum Numbers
{
    One = 1,
    Two = (MutantControl.IsActive(0) ? (One - 1) : (One + 1))
}
```

✗ This cannot compile since `MutantControl.IsActive(0)` is not a constant value.



How do we get the files?

We currently know:

- What mutation testing is, testing the tests
- Which strategies exist (mutant schemata best)
- How to place mutants using the Roslyn API

Project components

🎬 Project component

📁 Folder composite

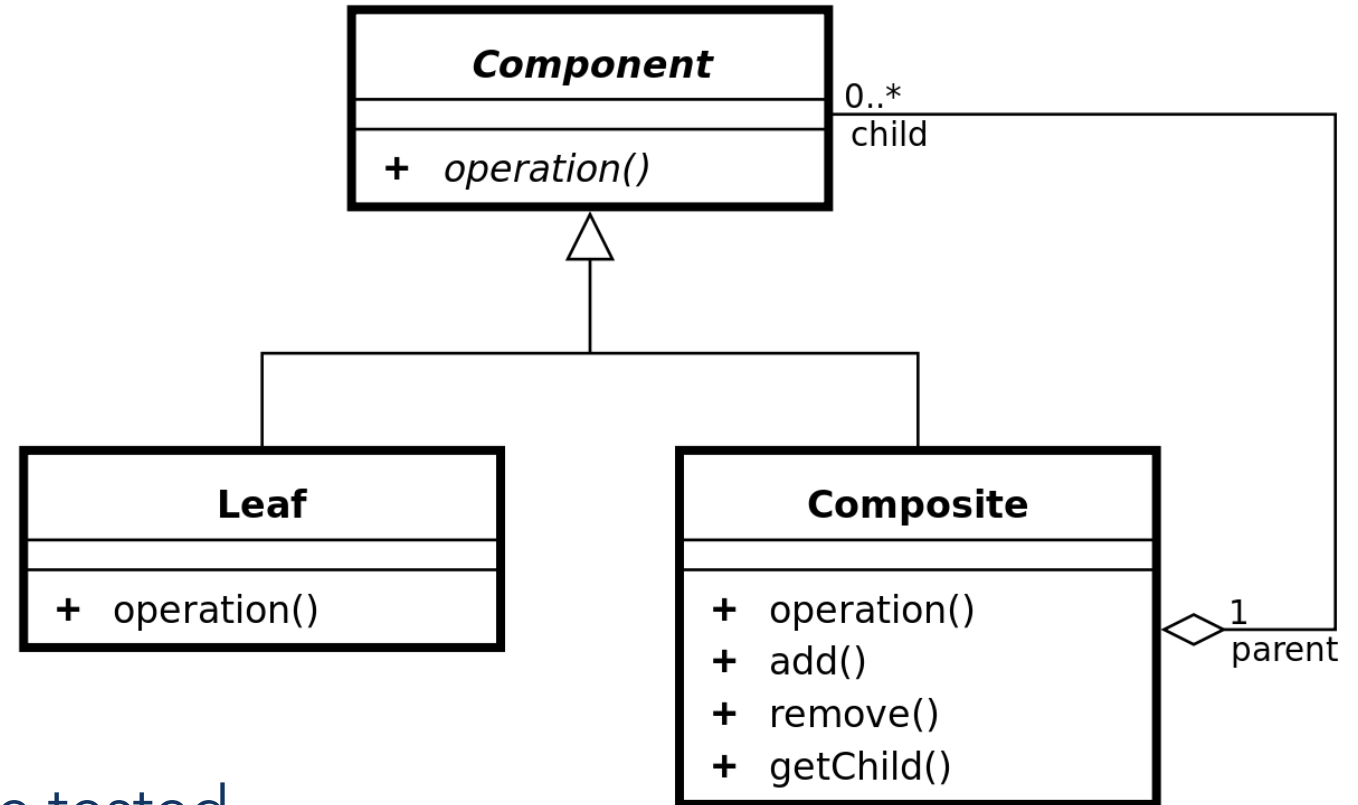
📄 File leaf

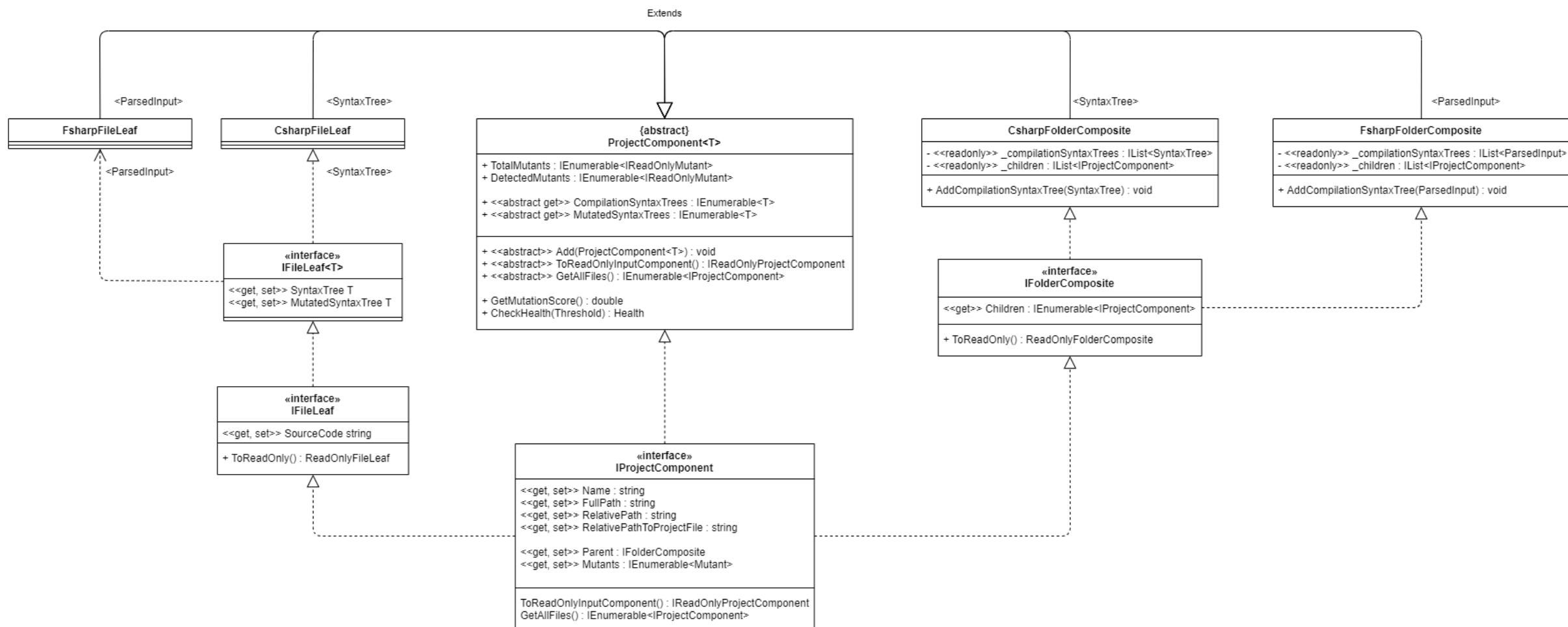
✗ F# syntax tree

🏗️ Differentiate C# & F#

✗ Not everything needs to be tested

🏗️ Read only







How do we run the tests?

We currently know:

- What mutation testing is, testing the tests
- Which strategies exist (mutant schemata best)
- How to place mutants using the Roslyn API
- How to get the files from the file system

▲ VSTest



Running tests



Collect diagnostics



Report results (we use our own)



Supports:

- XUnit
- NUnit
- MSTest

HTML Reporter

All files - Stryker.NET Report

All files

File / Directory	Mutation score	# Killed	# Survived	# Timeout	# No coverage	# Runtime errors	# Compile errors	Total detected	Total undetected	Total mutants
📁 All files	50.00% 50.00	2	2	0	0	0	0	2	2	4
📄 Properties/AssemblyInfo.cs	100.00% 100.00	0	0	0	0	0	0	0	0	0
📄 Person.cs	66.67% 66.67	2	1	0	0	0	0	2	1	3
📄 Program.cs	0.00% 0.00	0	1	0	0	0	0	0	1	1

Conclusion

- Stryker is a family of mutation testing frameworks
- Open source and maintained by InfoSupport
 - View our issues board if you want to contribute as well!
- We currently know:
 - What mutation testing is, testing the tests
 - Which strategies exist (mutant schemata best)
 - How to place mutants using the Roslyn API
 - How to get the files from the file system
 - How we run the tests with the VSTest API

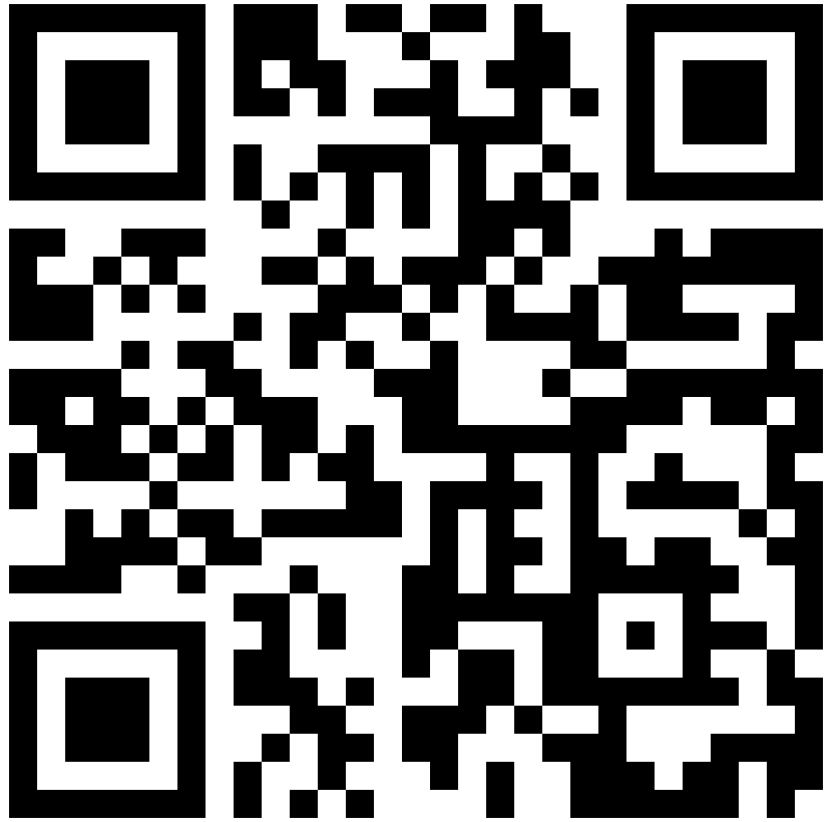


Questions?

Ossama.Sijbesma@infosupport.com



▲ Repository for labs and slides



[https://github.com/
OssamaSijbesma/stryker-labs](https://github.com/OssamaSijbesma/stryker-labs)



▲ Exercise 1 – Make it work

1. Install Stryker.NET (globally or check docs for local)

```
.NET CLI  
dotnet tool install -g dotnet-stryker
```

2. Navigate to your test folder

3. Run Stryker.NET and inspect the report

```
.NET CLI  
dotnet stryker -o
```





Exercise 2 - Improve mutation score

1. Fix the tests to improve your mutation score

All files - Stryker.NET Report



All files

File / Directory	 Mutation score	# Killed	# Survived	# Timeout	# No coverage	# Ignored	# Runtime errors	# Compile errors	Total detected	Total undetected	Total mutants
 All files	<div><div>59.26%</div></div> 59.26	16	9	0	2	0	0	0	16	11	27
 Basic.cs	<div><div>58.33%</div></div> 58.33	7	5	0	0	0	0	0	7	5	12
 Prime.cs	<div><div>60.00%</div></div> 60.00	9	4	0	2	0	0	0	9	6	15

▲ Repository for Stryker.NET



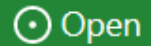
[https://github.com/
stryker-mutator/stryker-net](https://github.com/stryker-mutator/stryker-net)



Exercise 3 – Help me create a mutant

1. Fork Stryker.NET and inspect the project
2. Work out the mutator idea

Mutator idea: `string.IsNullOrEmpty` #1581



richardwerkman opened this issue on May 28, 2021 · 1 comment



richardwerkman commented on May 28, 2021

Member



Is your feature request related to a problem? Please describe.
When using `string.IsNullOrEmpty` most of the times only null is tested.

Describe the solution you'd like

If we mutate `string.IsNullOrEmpty(x)` into `(x != null)` and `(x != string.Empty)` it's clear both are tested.

▲ You can also find us @

01001001 01000011 01010100
MEET **Y**  **U**R FUTURE
01001001 01000011 01010100



