Clean as You Code

use Sonar and Roslyn analyzers to focus on the code you modify

Andrei Epure

Developer

Engineering Manager at sonar 

❤️ clean code & team work

Agenda

Why Clean Code is important

Roslyn and Sonar: tools for Clean Code

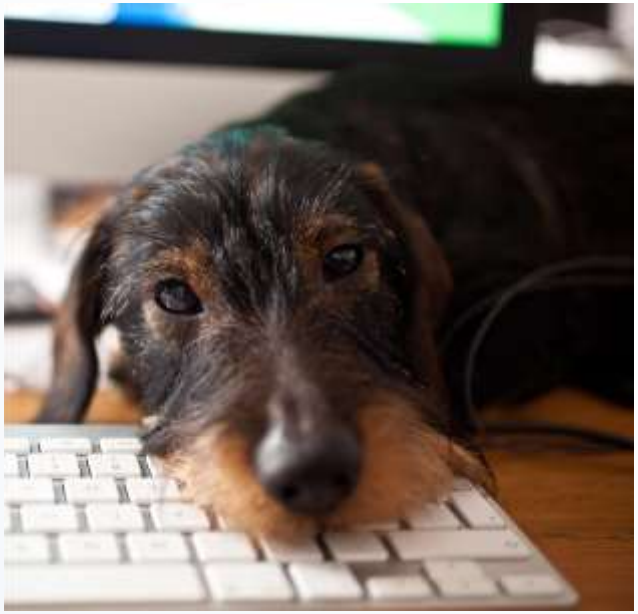
Clean as You Code

My experience at Sonar

Tim

Junior developer

Tired of long feedback loops



© Håkan Dahlström

Helen

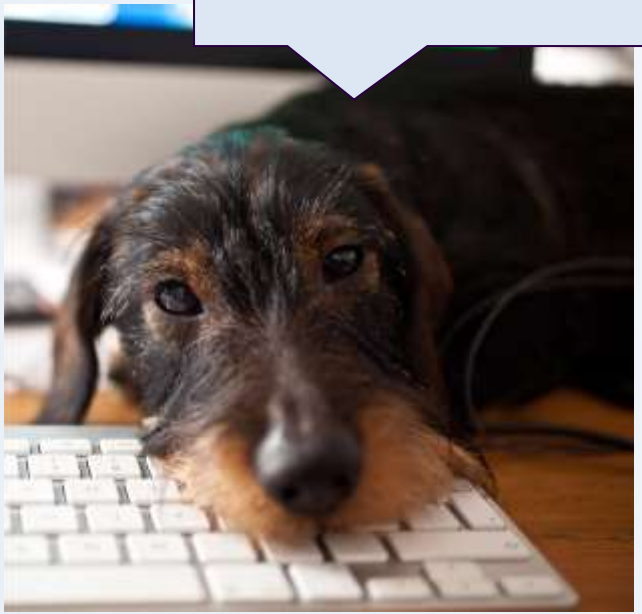
Senior developer

Quality gatekeeper

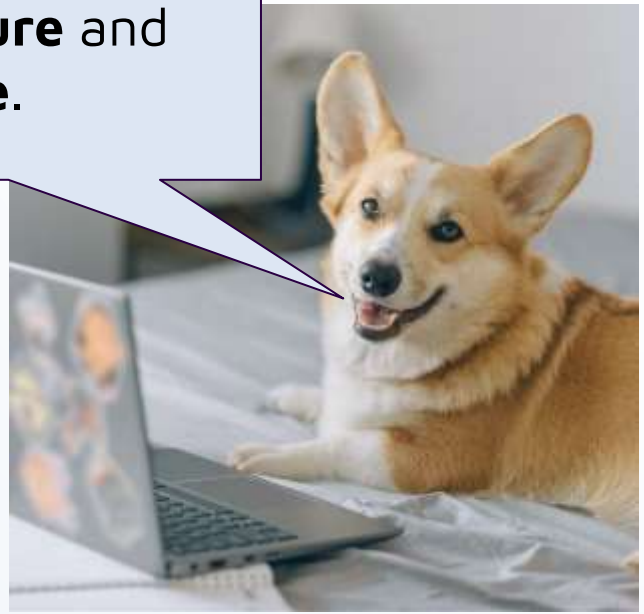
Busy

© Nataliya Vaitkevich

Helen, why do we need **Clean Code**?



Because we want our **software** to be **reliable**, **secure** and **maintainable**.



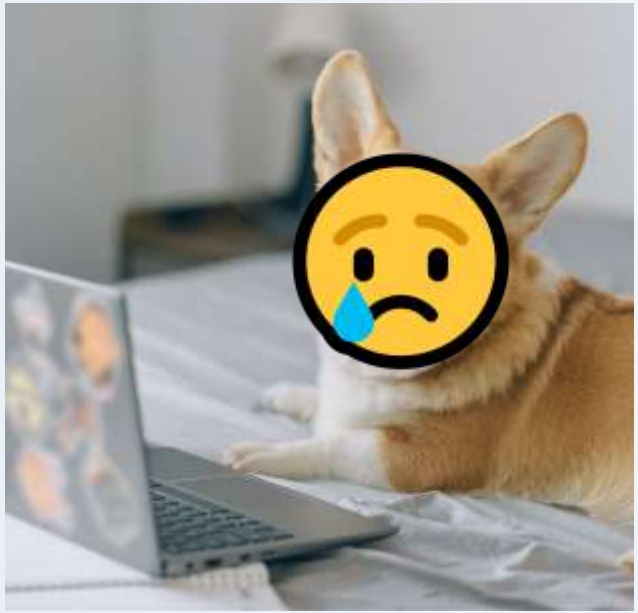
Clean Code
Attributes (☝guide)

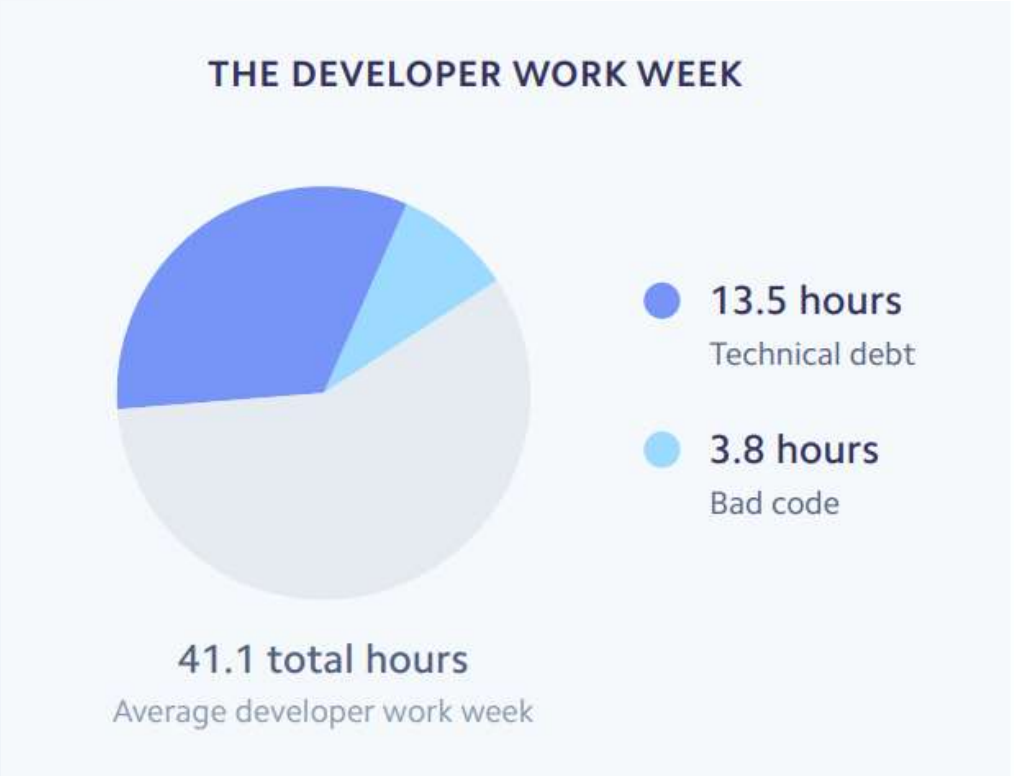
Software
quality

Consistent
Intentional
Adaptable
Responsible

Reliable (no bugs)
Secure (no vulnerabilities)
Maintainable (no code smells)
...

Clean Code:
development and production





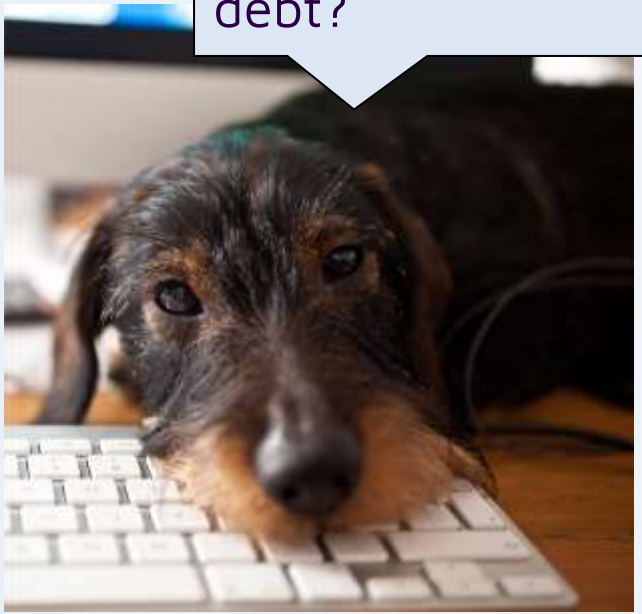
https://stripe.com/files/reports/the-developer-coefficient.pdf

"90% of reported security incidents result from exploits against defects in the design or **code** of software."

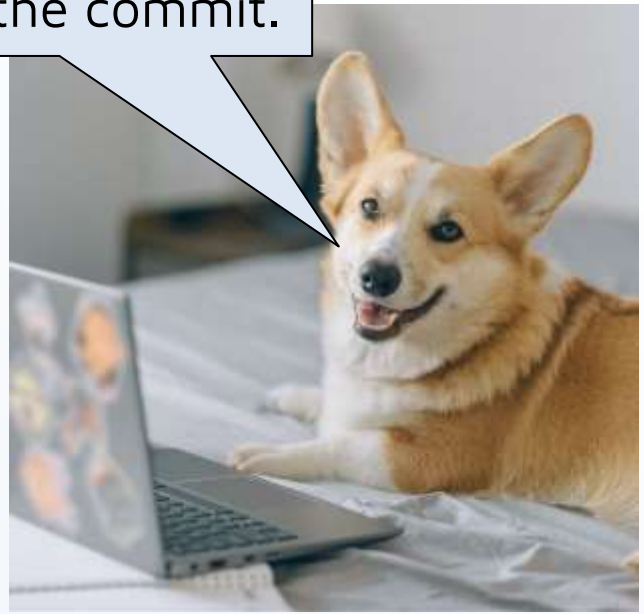
(U.S. Dept. of Homeland Security)

https://www.cisa.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf

Helen, why is there so much technical debt?



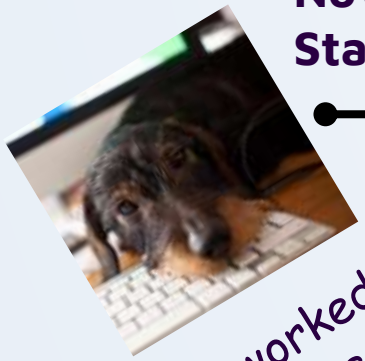
Our codebases are the best we could do on the day of the commit.



Over time, you will learn to improve your standards.

Novice Standard

Professional Standard



It worked on my machine

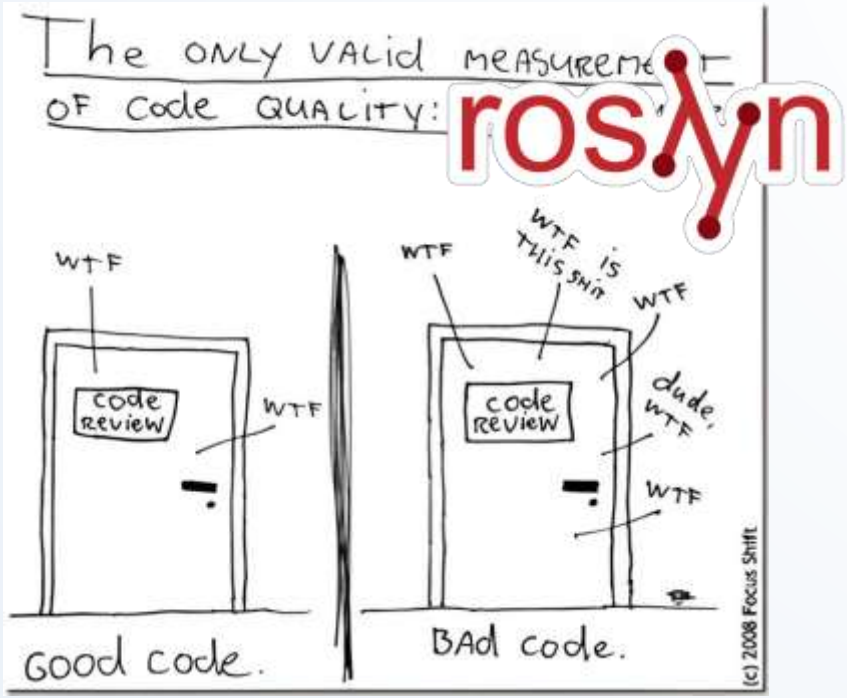
Clean Code

Helen 10 years ago
Standard

Helen Today
Standard

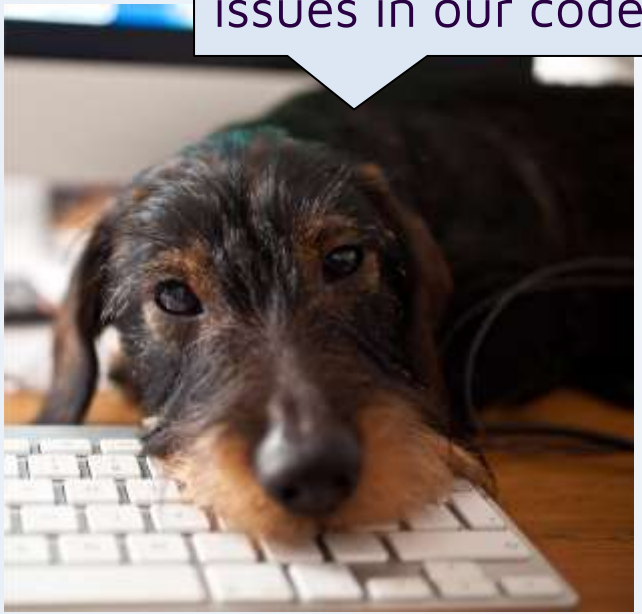
Clean Code

Measure Clean Code?

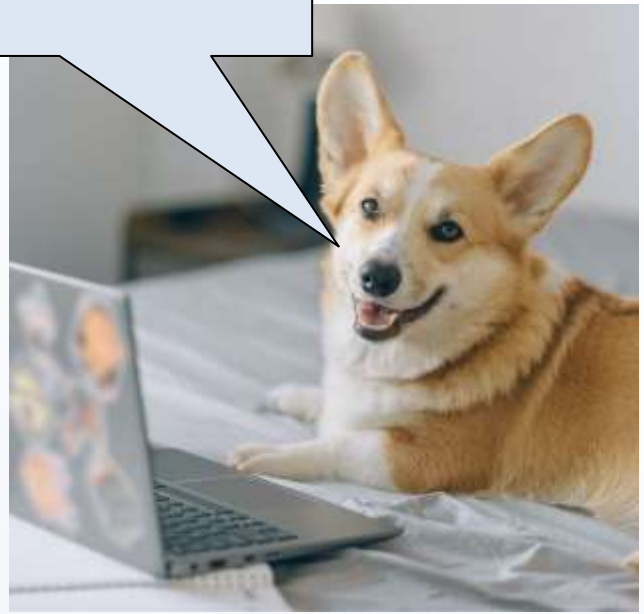


https://www.osnews.com/story/19266/wtfsm/

Helen, how do Roslyn analyzers find issues in our code?



They use static code analysis.



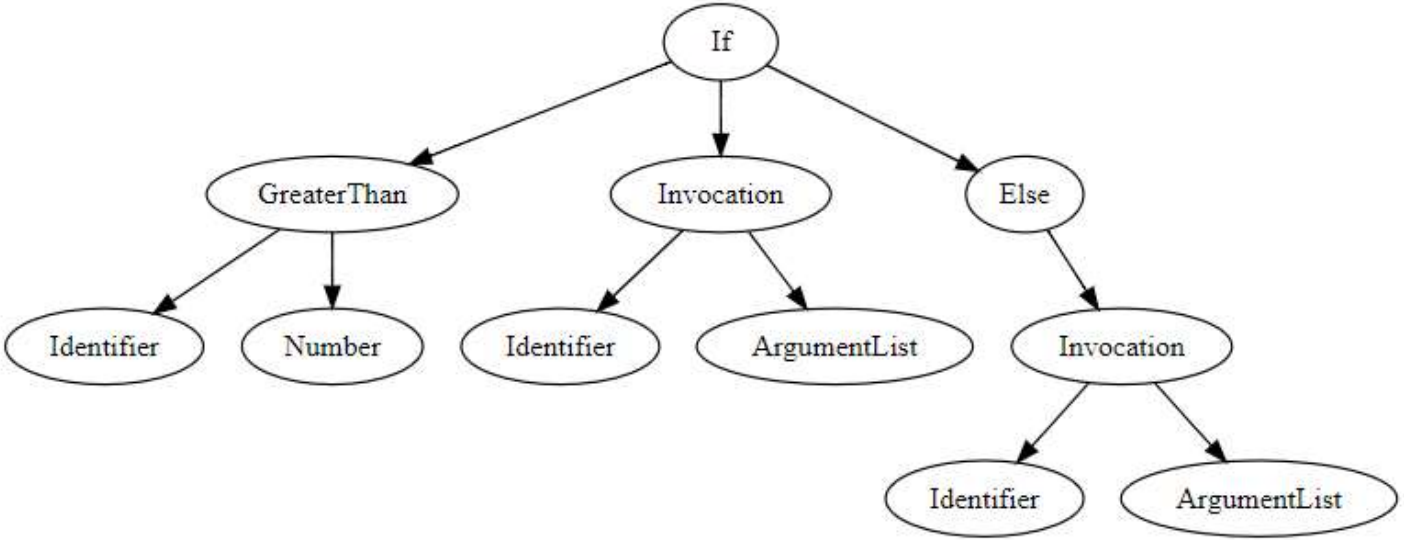
Static Analysis

Compiler framework for .NET

APIs for analyzing code **without** executing it

Syntax Tree

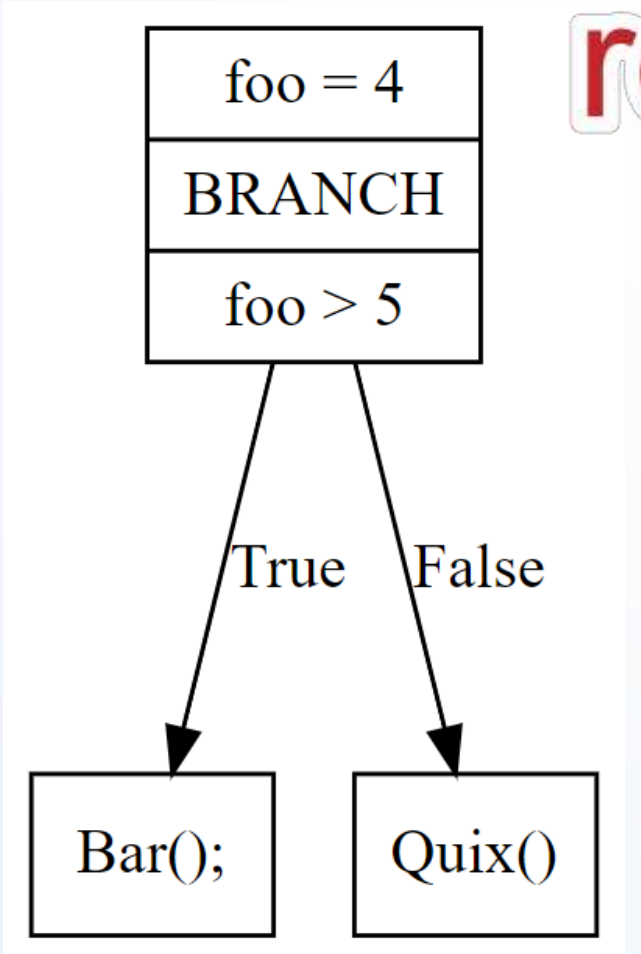
```
if (foo > 5)
    Bar();
else
    Quix();
```



https://edotor.net/

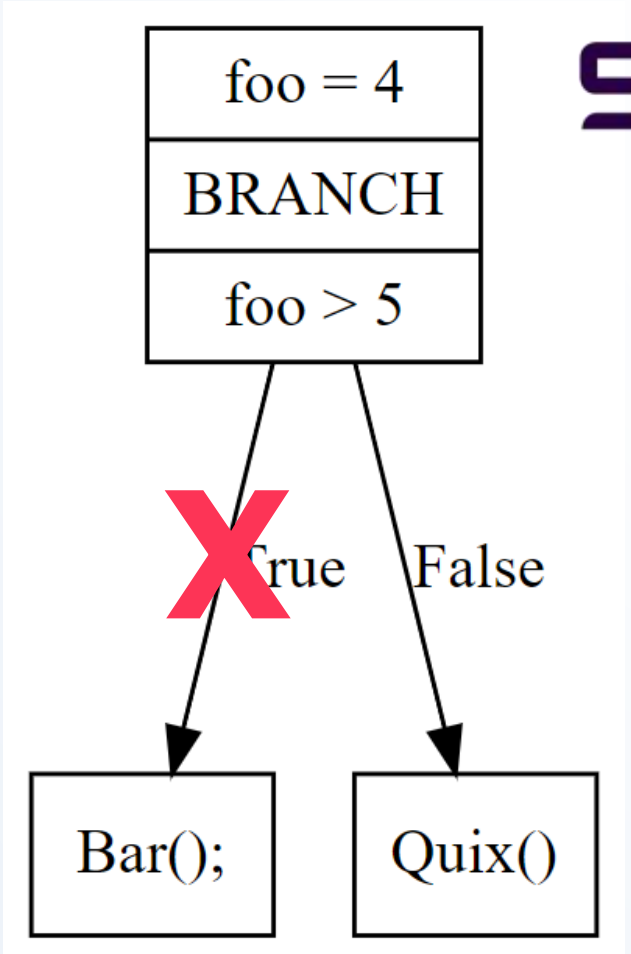
Control Flow Graph

```
var foo = 4;
if (foo > 5)
    Bar();
else
    Quix();
```



Symbolic Execution

```
var foo = 4;
if (foo > 5)
    Bar();
else
    Quix();
```



Symbolic Execution Example

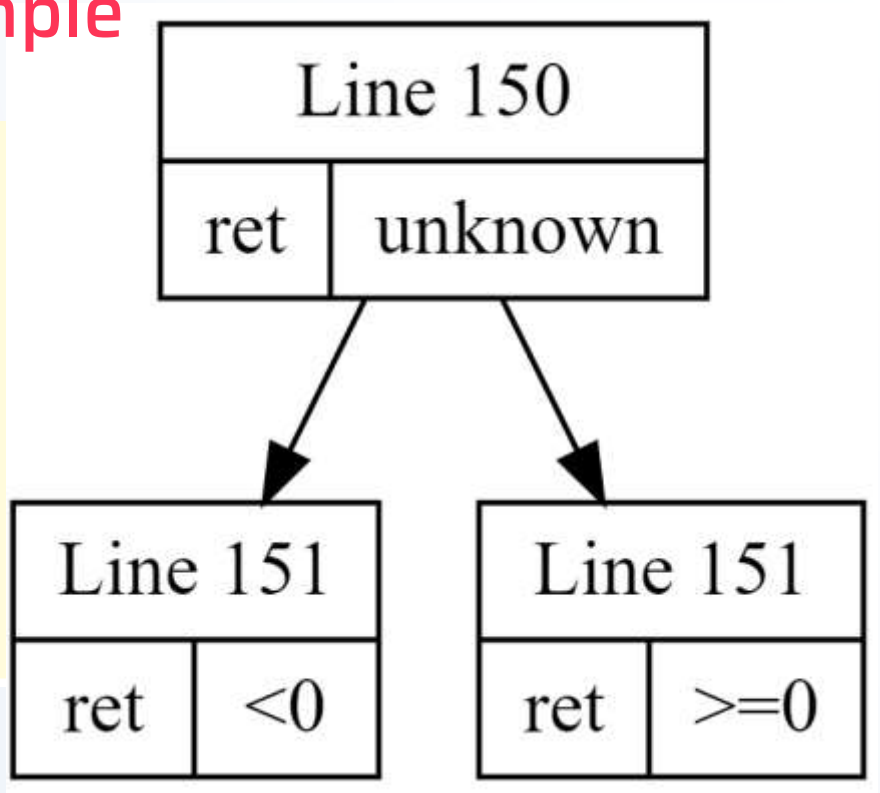
```
145         public long ToInt64()
146         {
147             if (IsNull)
148                 throw new SqlNullValueException();
149
150             long ret = _value / (s_lTickBase / 10);
151             bool fPositive = (ret >= 0);
152             long remainder = ret % 10;
153             ret /= 10;
154
155             if (remainder >= 5)
156             {
157                 if (fPositive)
158                     ret++;
159                 else
160                     ret--;
161             }
162
163             return ret;
164         }
```

SQLMoney.cs (dotnet/runtime)

https://github.com/dotnet/runtime/blob/45acd38/src/libraries/System.Data.Common/src/System/Data/SQLTypes/SQLMoney.cs#L150-L161 (MIT License)

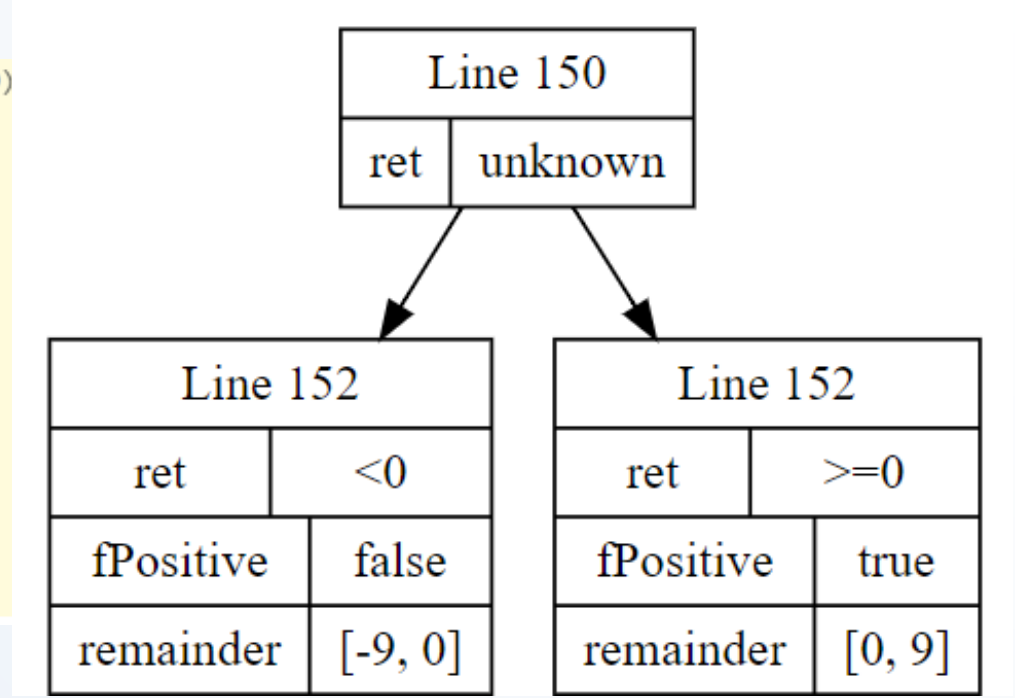
Symbolic Execution Example

```
150                 long ret = _value / (s_lTickBase / 10);
151                 bool fPositive = (ret >= 0);
152                 long remainder = ret % 10;
153                 ret /= 10;
154
155                 if (remainder >= 5)
156                 {
157                     if (fPositive)
158                         ret++;
159                     else
160                         ret--;
161                 }
```



Symbolic Execution Example

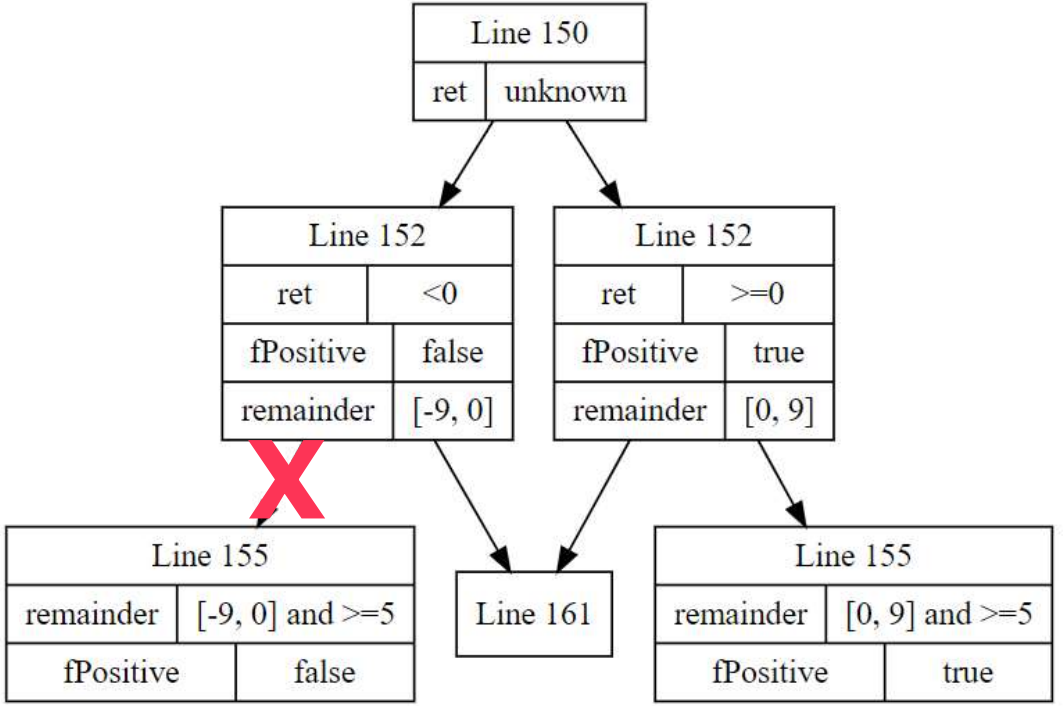
```
150                 long ret = _value / (s_lTickBase / 10)
151                 bool fPositive = (ret >= 0);
152                 long remainder = ret % 10;
153                 ret /= 10;
154
155                 if (remainder >= 5)
156                 {
157                     if (fPositive)
158                         ret++;
159                     else
160                         ret--;
161                 }
```



sharplab

Symbolic Execution Example

```
150                 long ret = _value / (s_lTickBase
151                 bool fPositive = (ret >= 0);
152                 long remainder = ret % 10;
153                 ret /= 10;
154
155                 if (remainder >= 5)
156                 {
157                     if (fPositive)
158                         ret++;
159                     else
160                         ret--;
161                 }
```



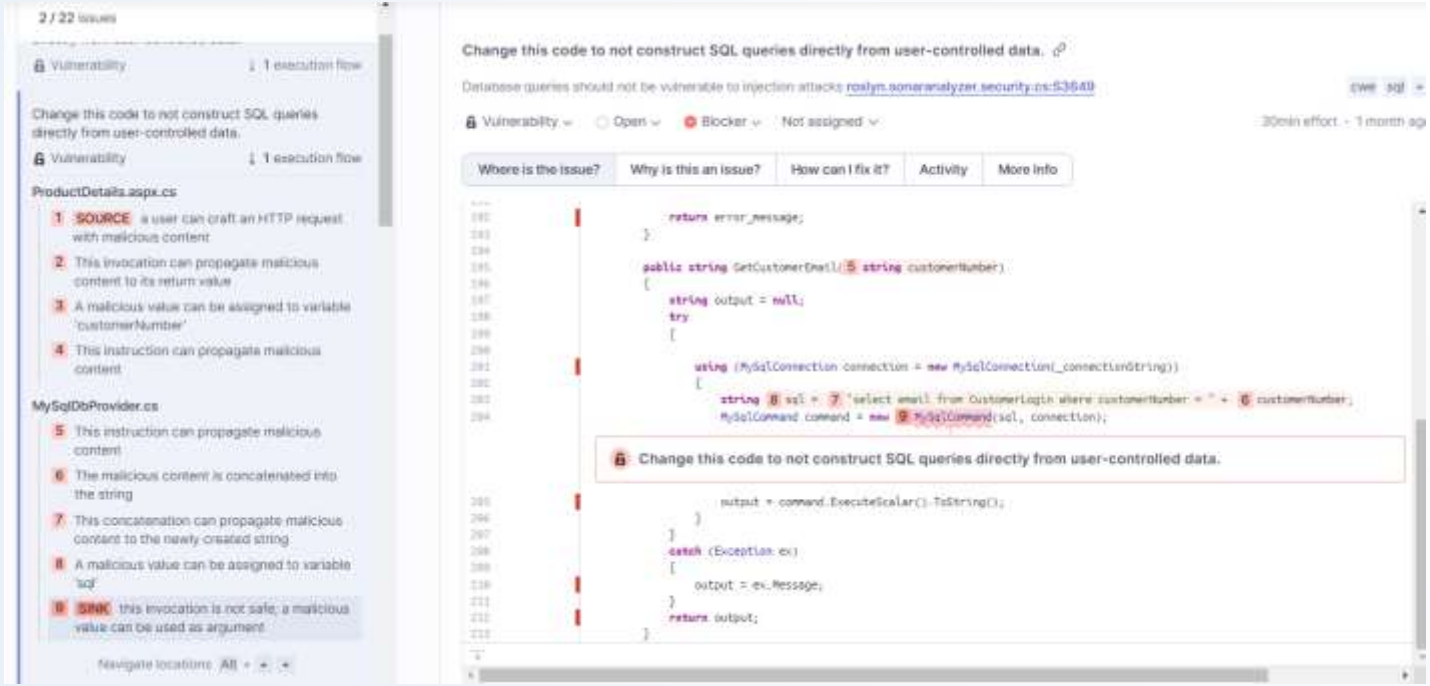
```
150             long ret = _value / (s_lTickBase / 10);
151             bool fPositive = (ret >= 0);
152             long remainder = ret % 10;
153             ret = ret / 10;
154
155             if (remainder >= 5)
156             {
157                 if (fPositive)
```

Change this condition so that it does not always evaluate to 'True'. Some code paths are unreachable.

```
158                     ret++;
159                 else
160                     ret--;
161             }
162
163             return ret;
```

- dotnet/runtime/issues/90741
- sharplab

Taint analysis



Tools

Who is using Roslyn analyzers?

Tools

Free analyzers on nuget.org

Sort(x => x.Downloads)

xUnit.Analyzers - 413M

StyleCop.Analyzers - 137M

Microsoft.Azure.Functions.Analyzers - 50M

Microsoft.VisualStudio.Threading.Analyzers - 45M

SonarAnalyzer.CSharp - 40M

Microsoft.CodeAnalysis.NetAnalyzers - 31M

Required config

xUnit.Analyzers - bundled with xUnit

StyleCop.Analyzers - you need to configure it

 SonarAnalyzer.CSharp  sonar

Microsoft.CodeAnalysis.NetAnalyzers - built in SDK starting .NET 5

Tools

Who is using Roslyn analyzers?

Setup  analyzers

With **Directory.Build.props** in the root of the solution

Configure rules with **.editorconfig**

Demo

roslyn side effects

Runs during the build → it will slow down the build

.NET SDK analyzers - 24 default rules

Sonar - 310 default rules → we made ☝ a performance guide

Tools

Developers write **clean** code

Teams have a **common** standard

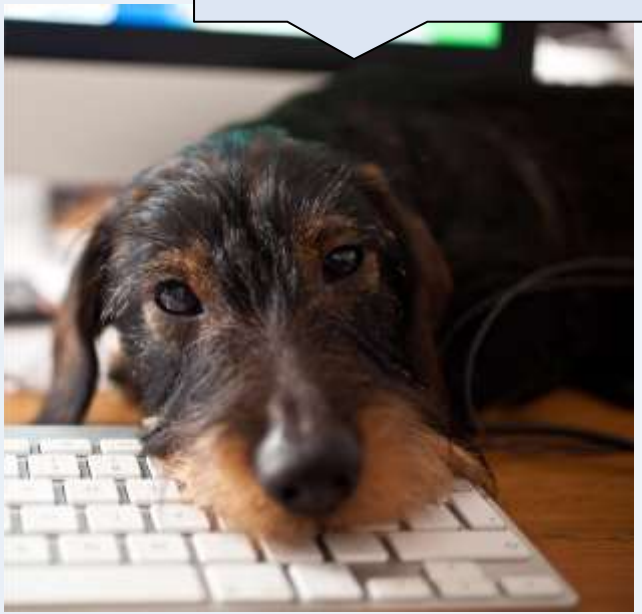
Measure Clean Code?

Roslyn Analyzers

Overview with ☝ Sonar

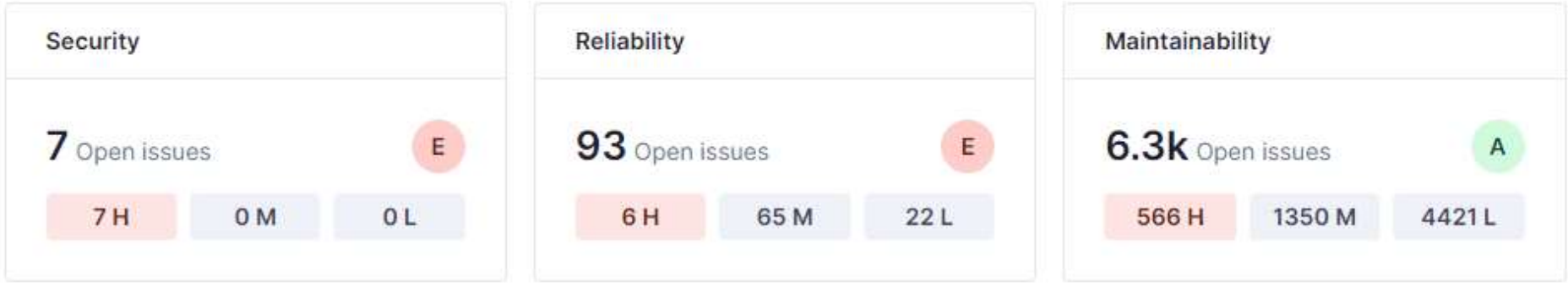
30 seconds break

These tools are awesome!



Yes, but knowing is not enough…

Knowing is not enough



How can we clean our codebase?

Challenges

Deliver new functionality

Risk of functional regression

It can be boring

Knowing is not enough

Things You Should Never Do

Option 1: The Rewrite

Knowing is not enough

Option 2: The big refactor

Knowing is not enough

Option 3: Clean as You Code

Clean as You Code

Focus on *New Code* : added or modified

Don’t (re)introduce new issues

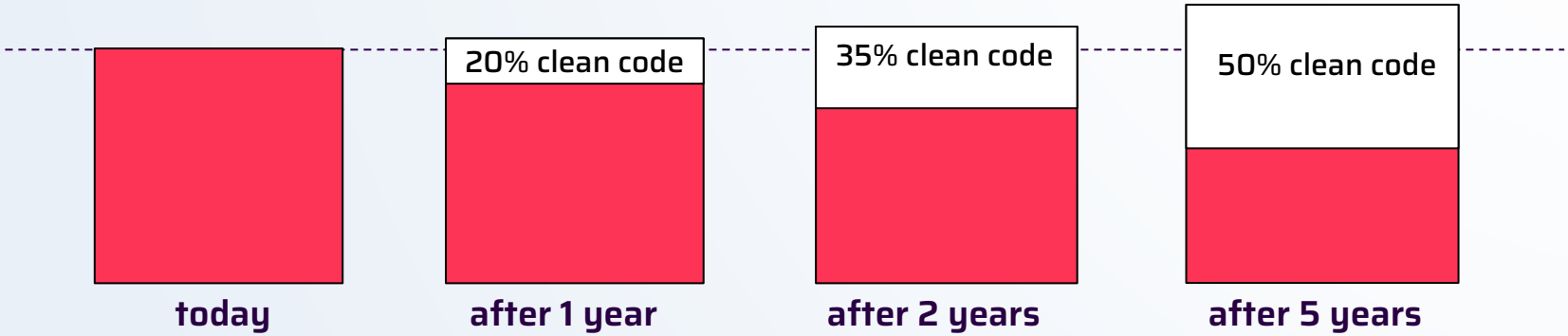
Clean as You Code

The code is fresh

The cost is ~0

Clean as You Code

Your existing codebase gets progressively clean

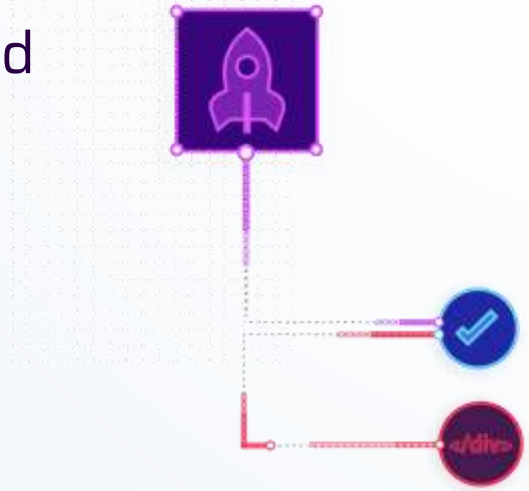


Implementing Clean as You Code

Set up a **Quality Gate** on **new code** based on your standard (**Quality Profile**)

Don't **merge** unless it is green

Don't **release** unless it is green



Implementing Clean as You Code

New Code Definition

- Pull Request / Commit
- Versions
- Number of days

Clean as You Code Demo

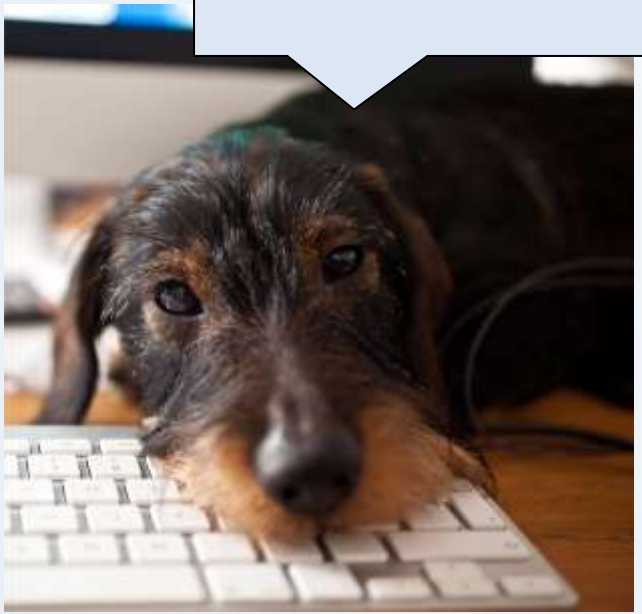
☝ Quality Gate (main branch)

Overall Code vs. New Code

☝ Pull Request integration

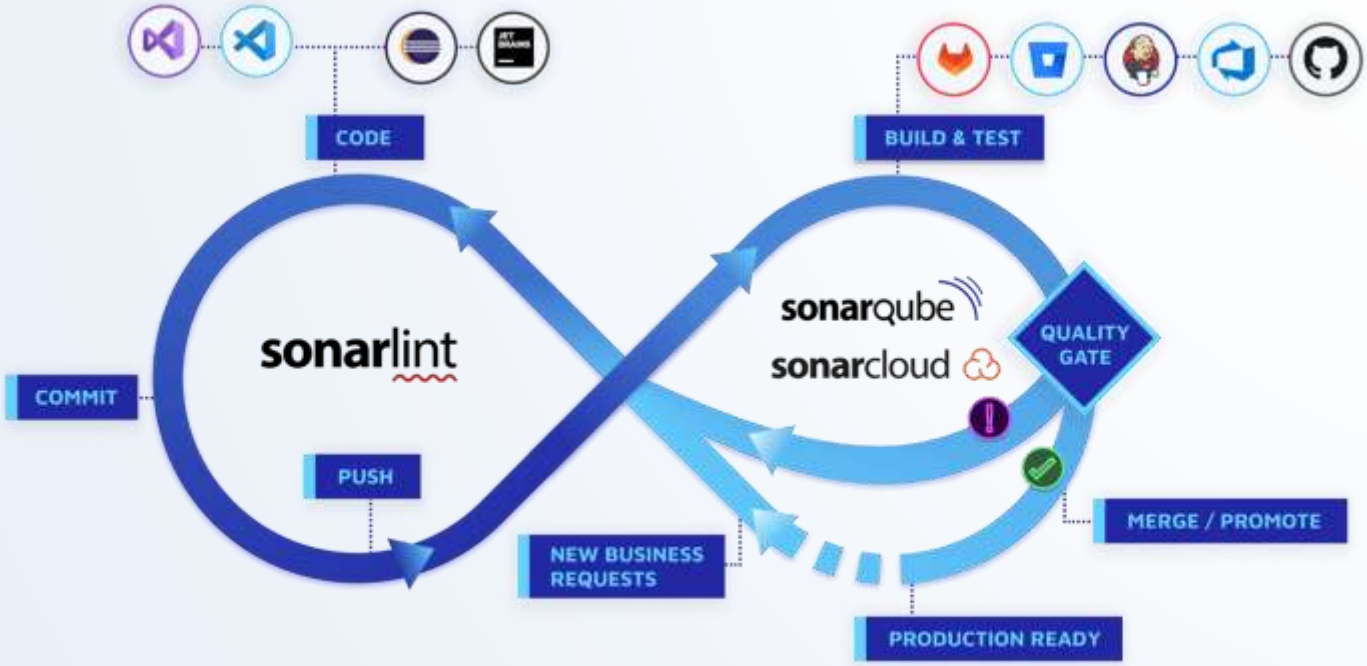
SonarLint

I learn as I code



I can focus on more important things during code reviews

Part of Development



Why does it work?

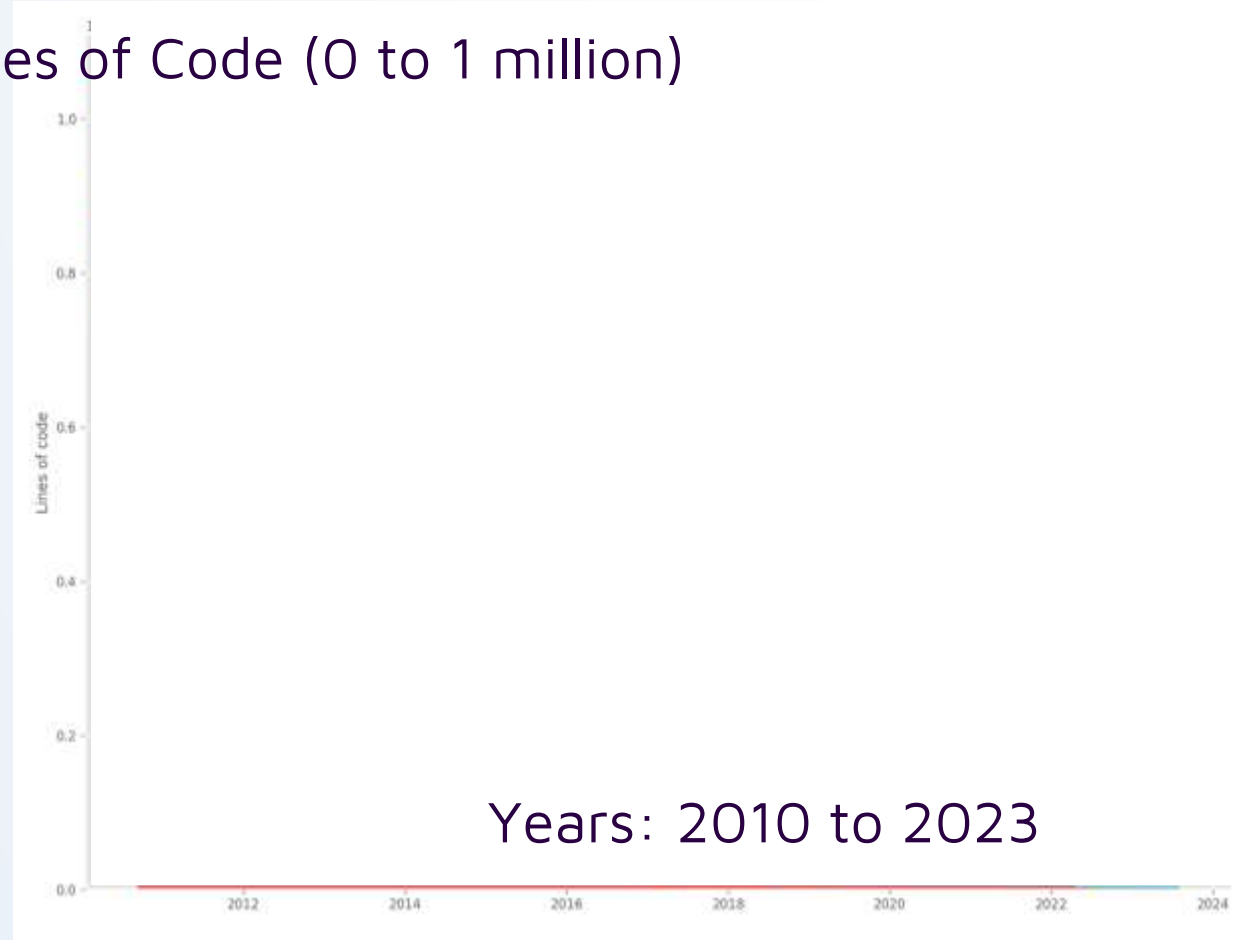
On average, 50% of code* gets changed within 3.33 years.

*of large open-source projects on GitHub

☝ https://github.com/erikbern/git-of-theseus

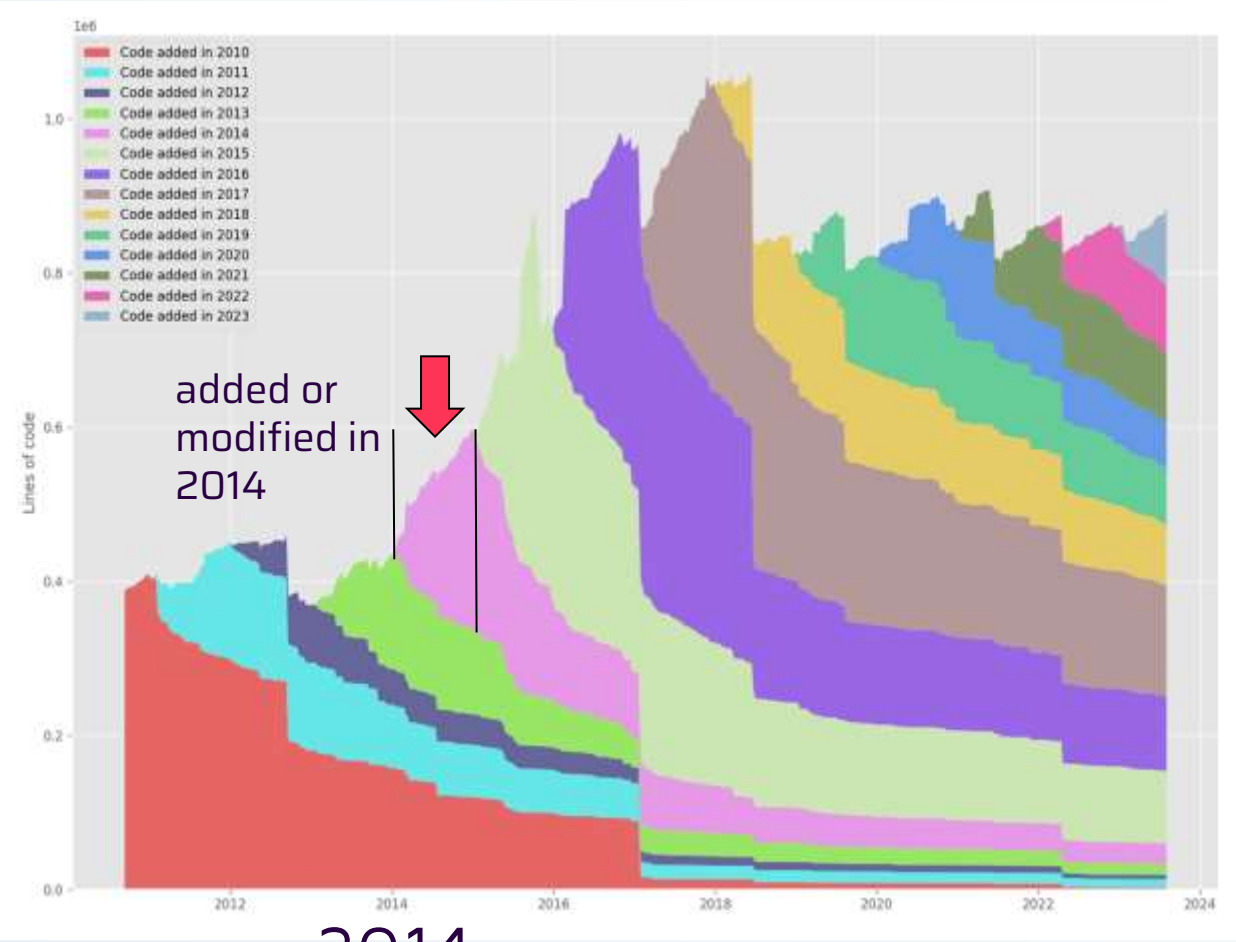
Here's SonarQube

Lines of Code (0 to 1 million)



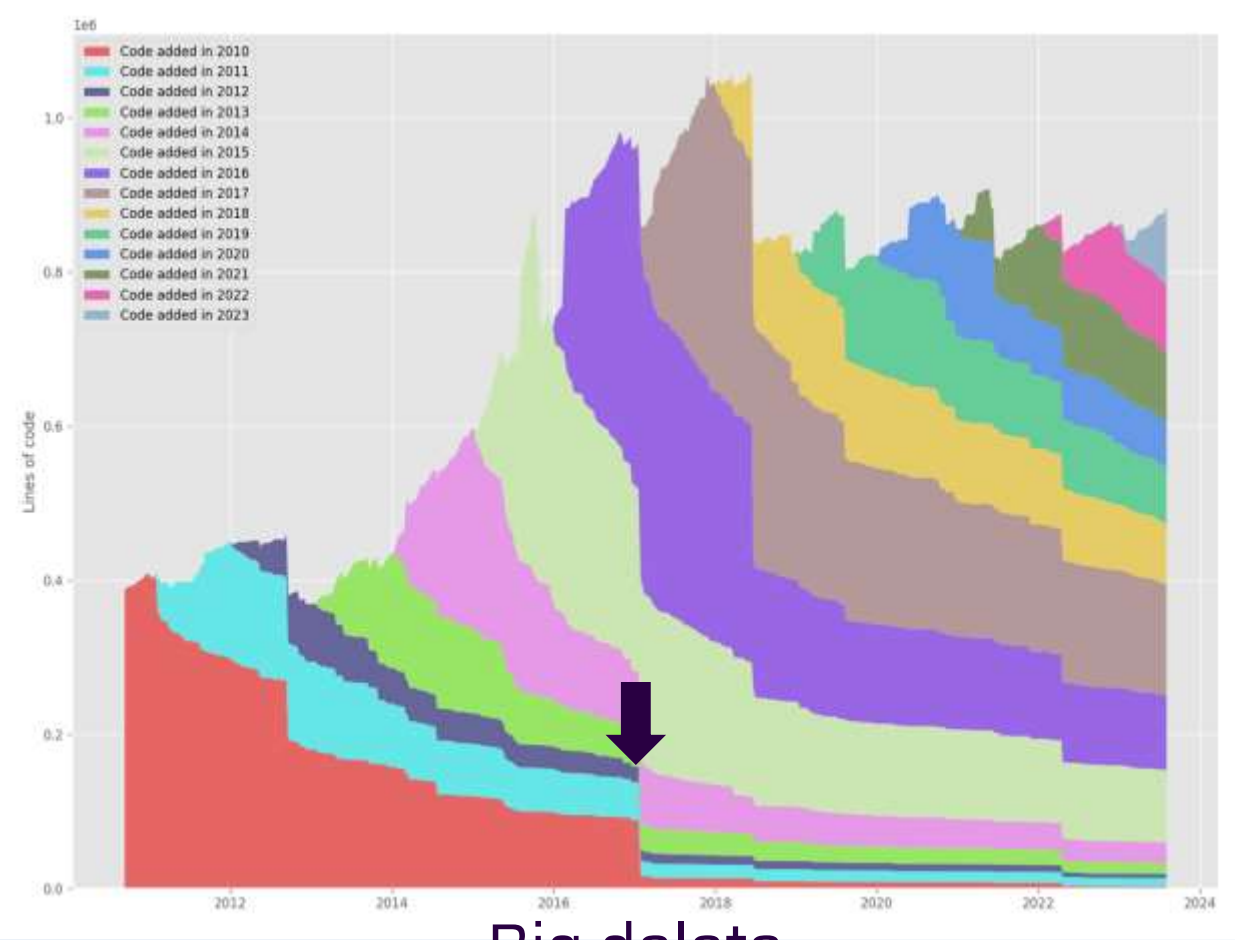
Years: 2010 to 2023

Here's SonarQube



2014

Here's SonarQube

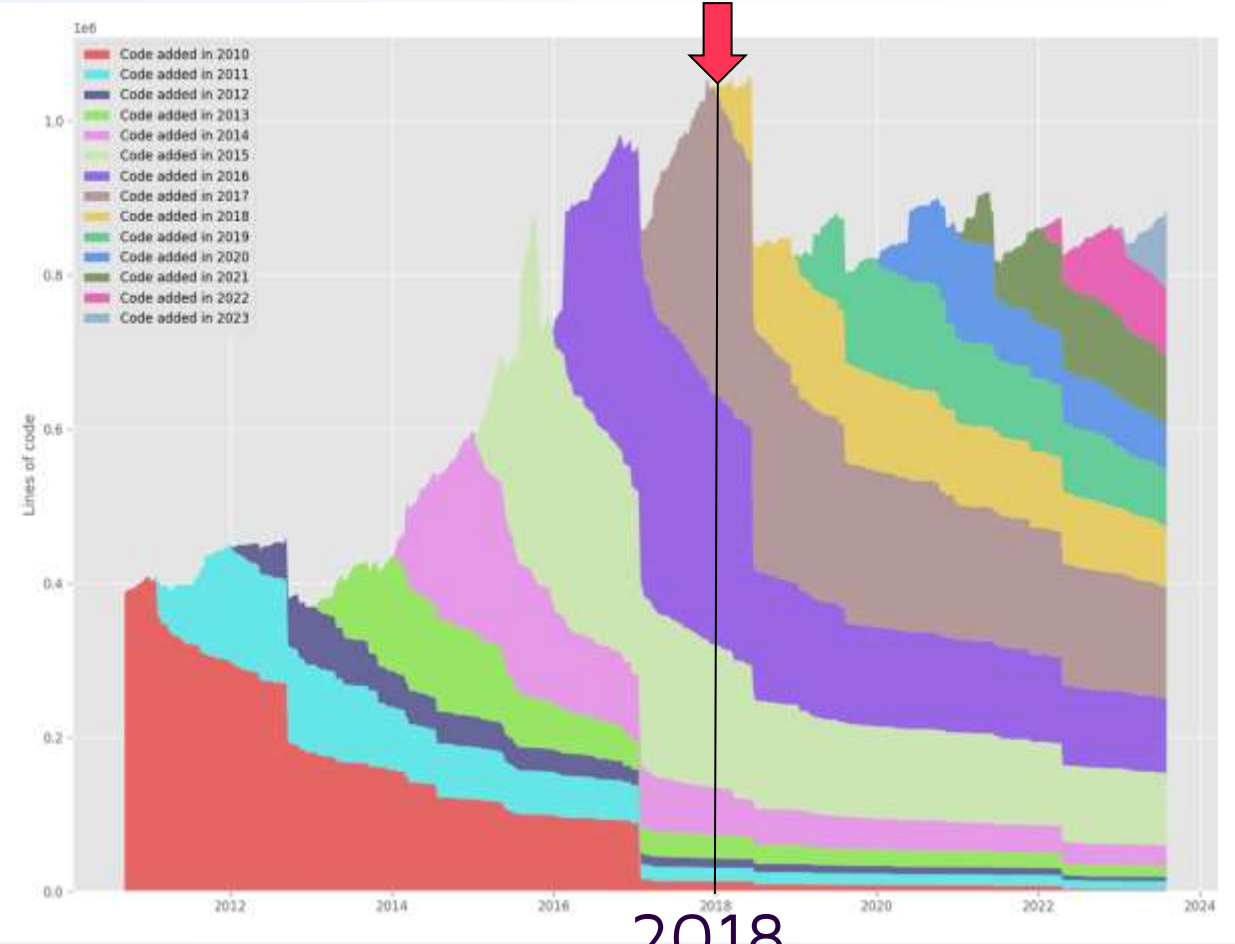


Big delete

(bye-bye ruby code)

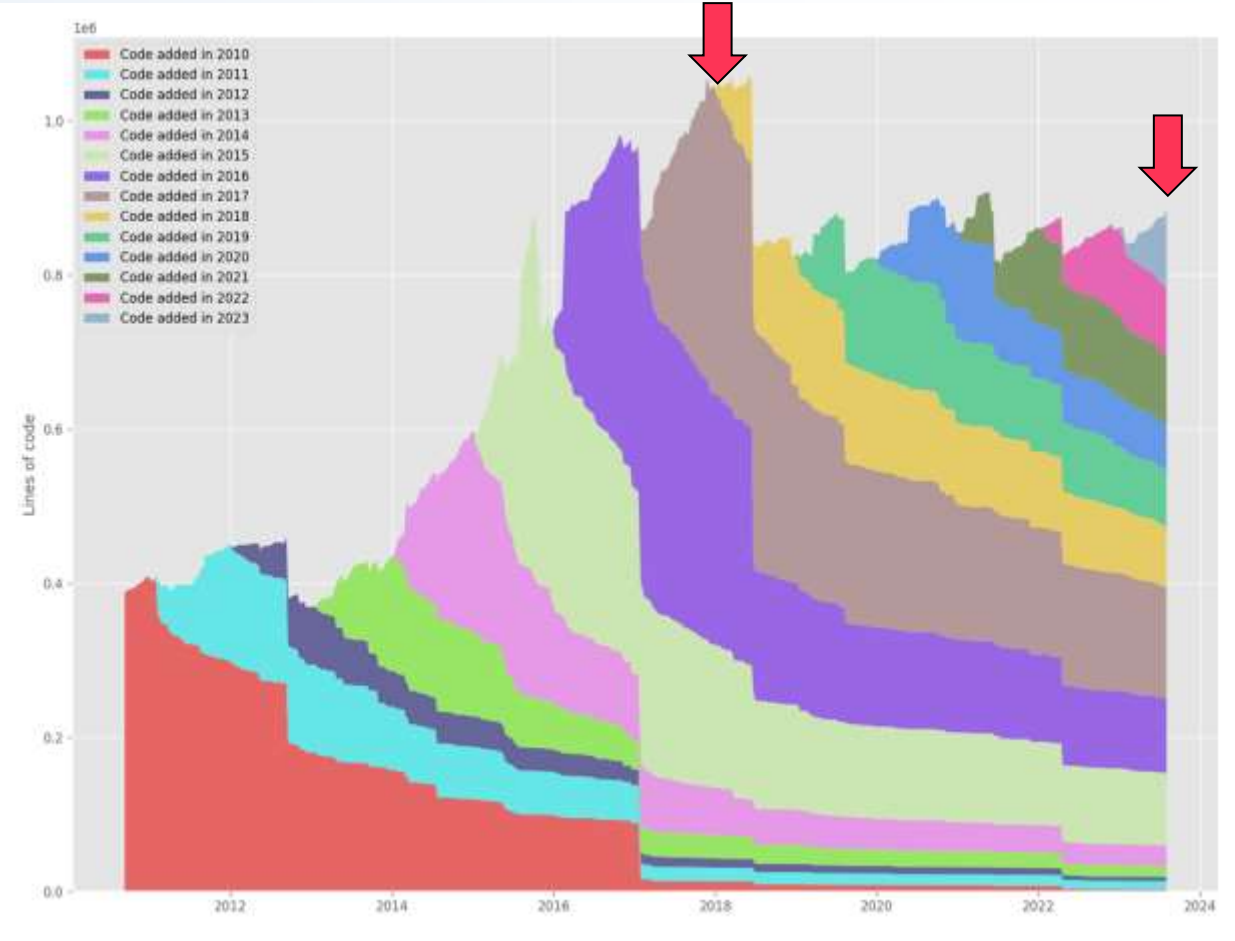
Here's SonarQube

At the beginning of 2018 there were 1 million LOC



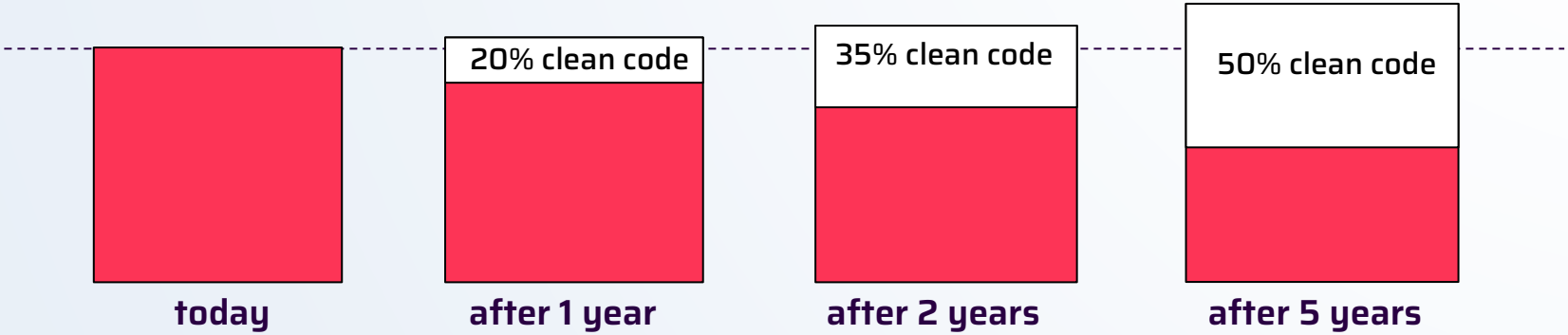
Here's SonarQube

Out of only 1 million LOC in 2018 less than 500K remain today



Clean as You Code

Your existing codebase gets progressively clean



My experience at Sonar

Quality Profile + ☝ StyleCop.Analyzers

Quality Gate (QG)

- New Code: 95% ccov and no issues
- Overall code: no major bugs/vulnerabilities

My experience at Sonar

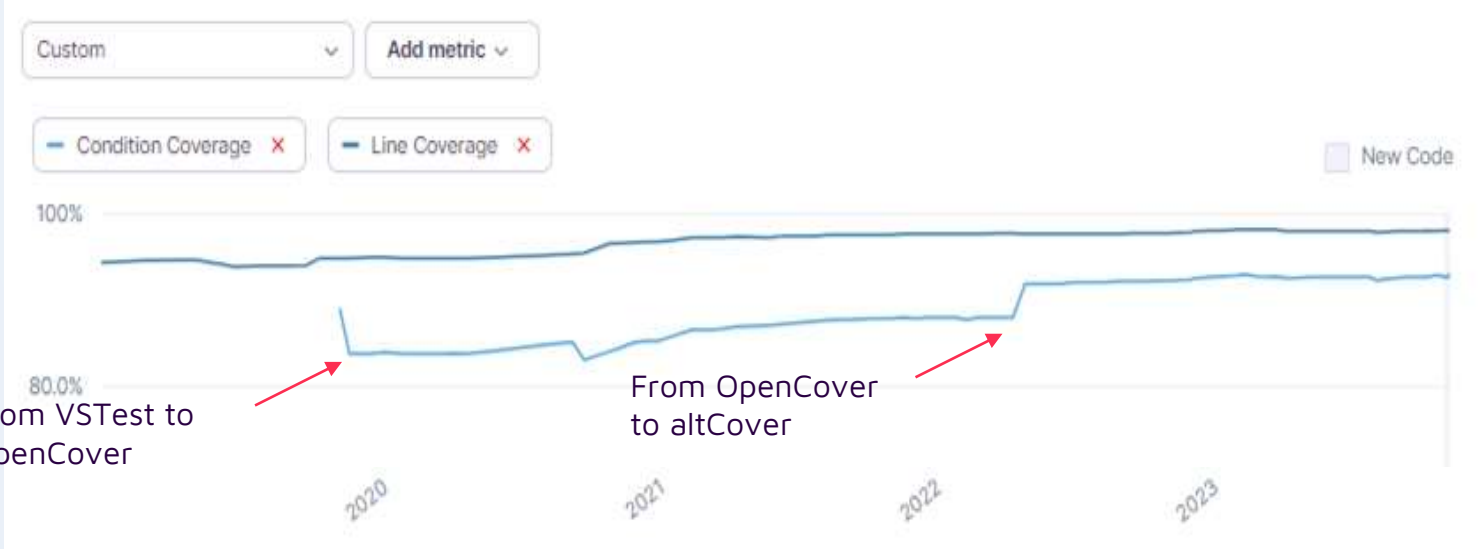
We don’t merge PRs with red QG

Red QG = broken build (slack notification)

My experience at Sonar

In three years, for ☝ sonar-dotnet, we increased branch (conditional) coverage from 82% to 93% by using a Quality Gate at 95%.

My experience at Sonar



```
if (foo > 5 && foo < 9)
    Bar();
```

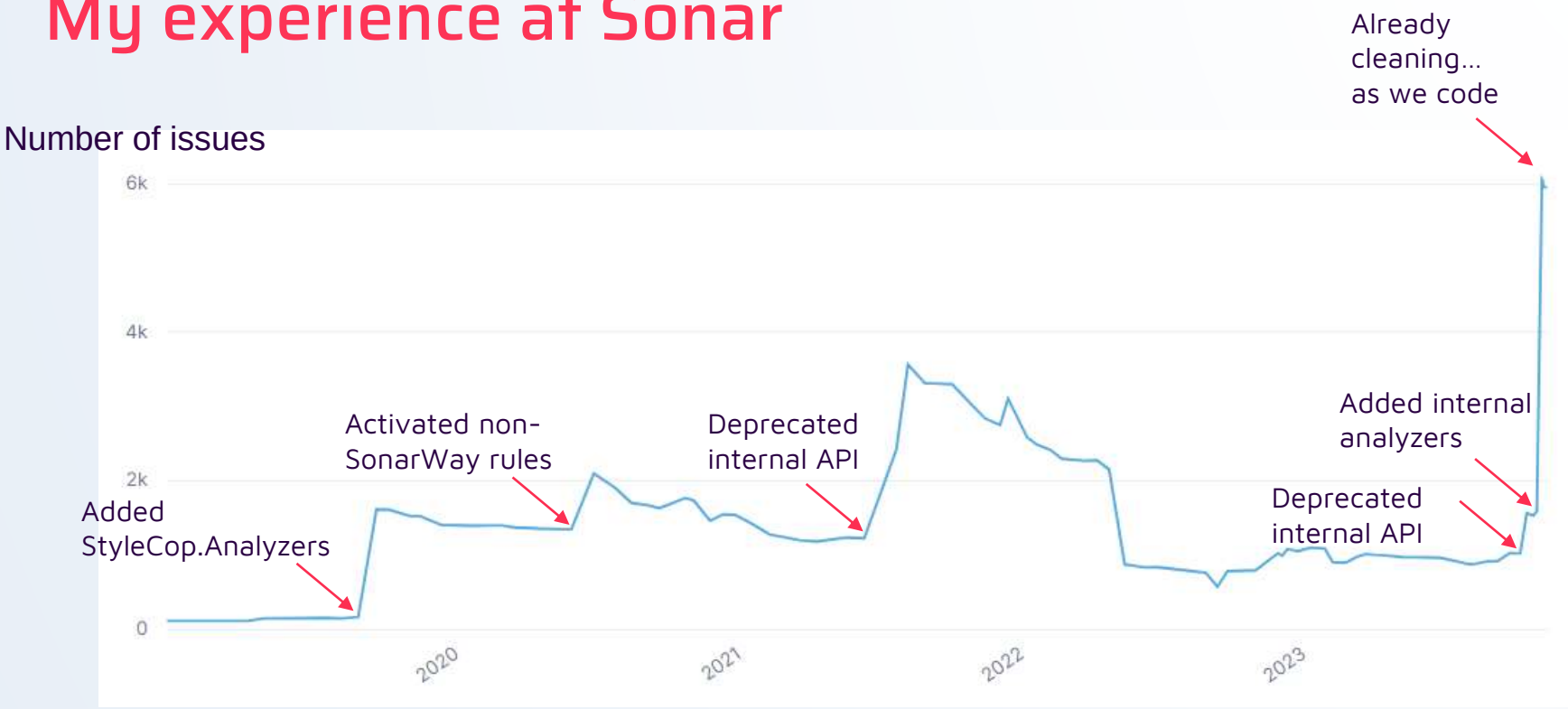

100% line coverage with testing when **foo** is 6

My experience at Sonar

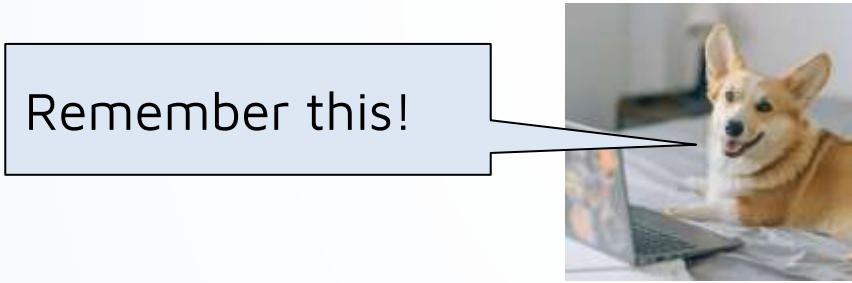
sonar-dotnet (SonarWay rules from SQ LTS 9.9)

Version (year)	Number of C# LOC (thousands)	Maintainability issues
8.1 (2019)	61 K	182
8.15 (2020)	65 K	184
8.31 (2021)	68 K	193
8.48 (2022)	78 K	160
9.13 (2023)	70 K	166

My experience at Sonar



Key takeaways



Clean as You Code = improve the code you touch:

- Set your **common** standard of **clean** code
- Ensure every commit achieves that standard
- Use static analysis to help consistently achieve it

Key takeaways

Free and open-source

Free

for open source

Community Edition

Import 3rd party Roslyn issues by default

Feedback form & slides:

 AndreiEpure.ro

sonar

Extra slides

Clean as You Code

Happy that Roslyn analyzers exist because GenAI will produce a lot of code.

sonar is more

470+ C# Rules

30+ languages

rules.sonarsource.com