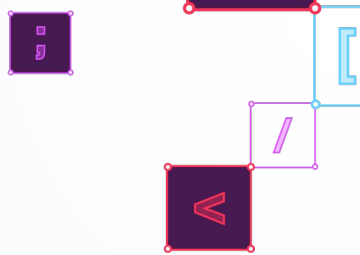
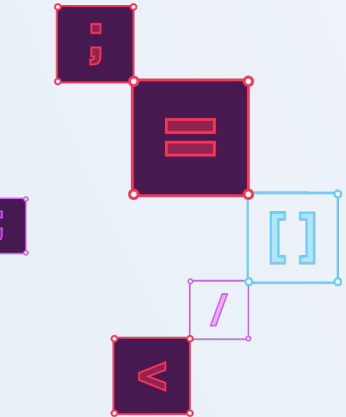


# Clean as You Code

## use Roslyn analyzers to focus on the code you modify



**Andrei EPURE**

**29.08.2023**



sonar

## Gold





## Silver


Digitec Galaxus AG

# Me - Andrei Epure

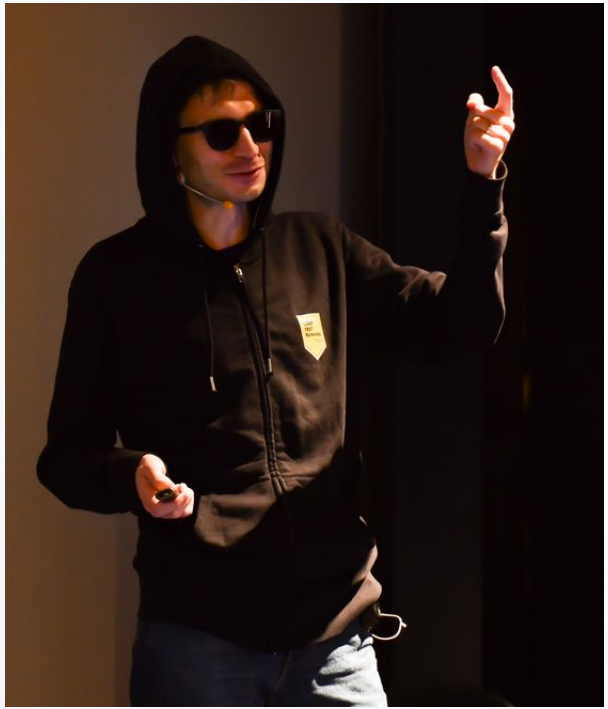
Developer

Engineering Manager at sonar

❤️ clean code & team work

# Me - Mr. Evil Hacker

.NET Day Switzerland 2022



# Agenda

Why is Clean Code important

Static Analysis

Clean as You Code

My experience at Sonar

Tim

Junior developer

Tired of long feedback loops

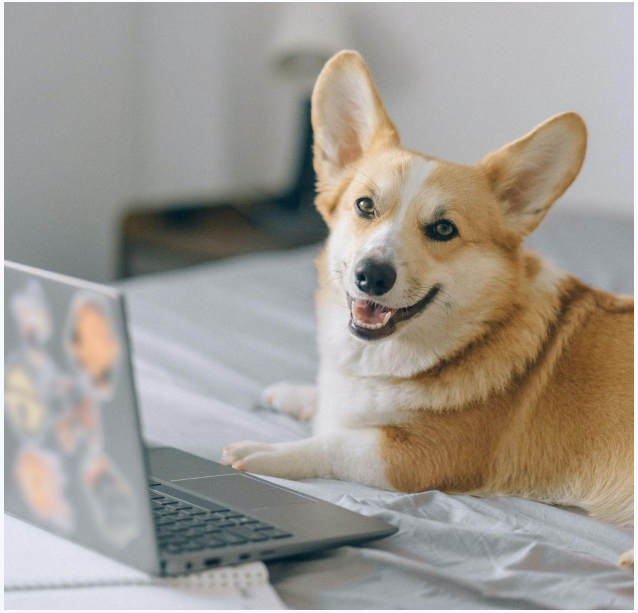
© Håkan Dahlström

Helen

Senior developer

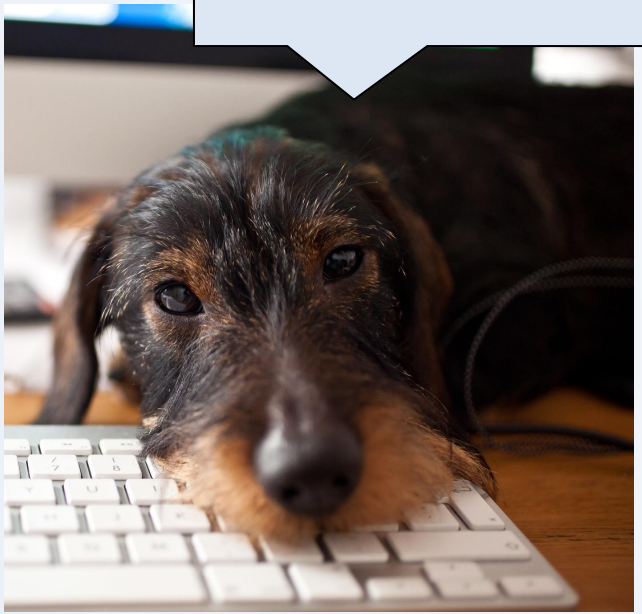
Quality gatekeeper

Busy

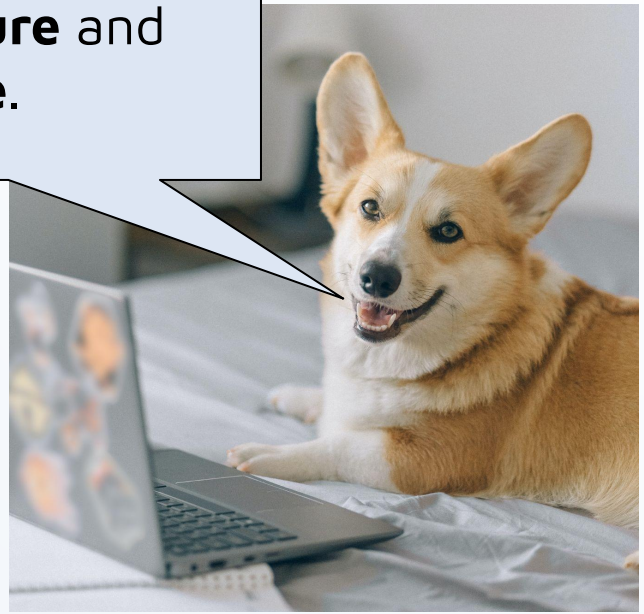


© Nataliya Vaitkevich

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

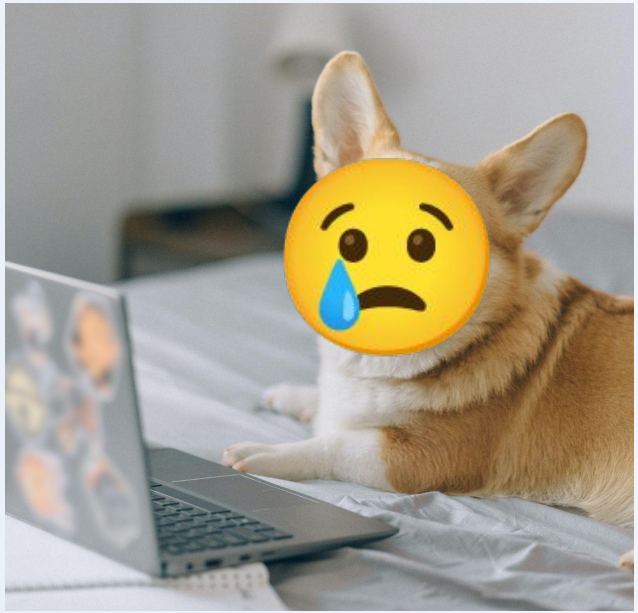


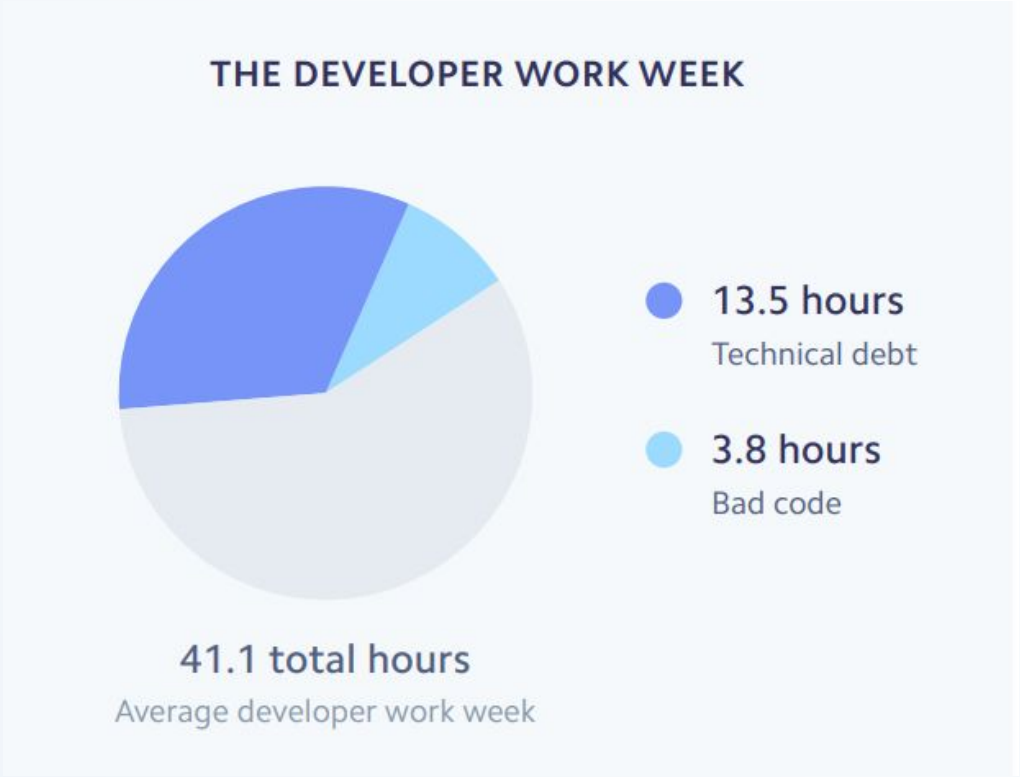
Because we want our **software** to be **reliable**, **secure** and **maintainable**.



# Why is Clean Code important for **you**?

For **me**:
development and production





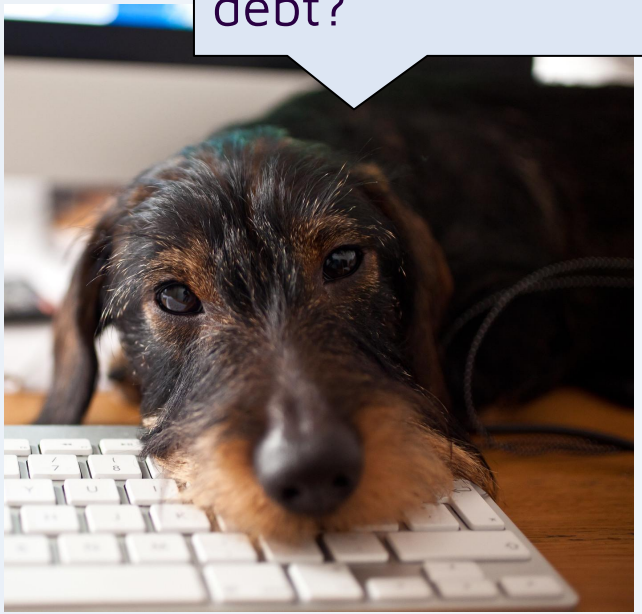

https://stripe.com/files/reports/the-developer-coefficient.pdf

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

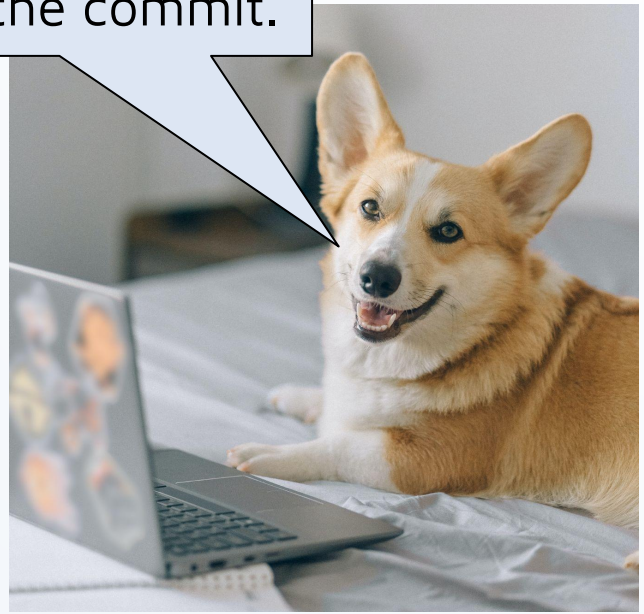
(U.S. Dept. of Homeland Security)

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.

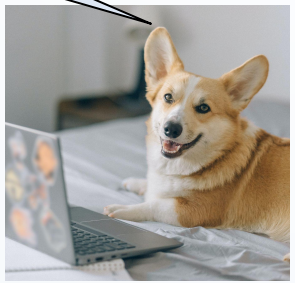


Over time, you will learn to improve your standards.

**Novice Standard**

It worked on my machine

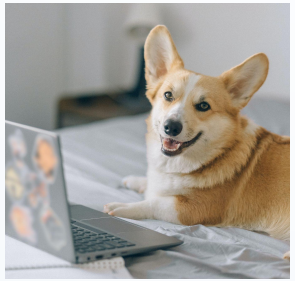
**Professional Standard**



# Clean Code

**Helen 10 years ago**
**Standard**

**Helen Today**
**Standard**

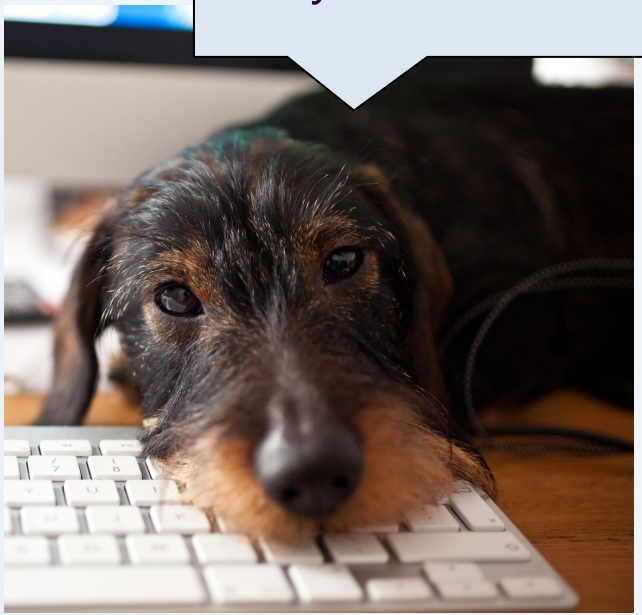


# Clean Code

© Cory Denton from Saskatoon

# Do a code review of your code 10 years ago

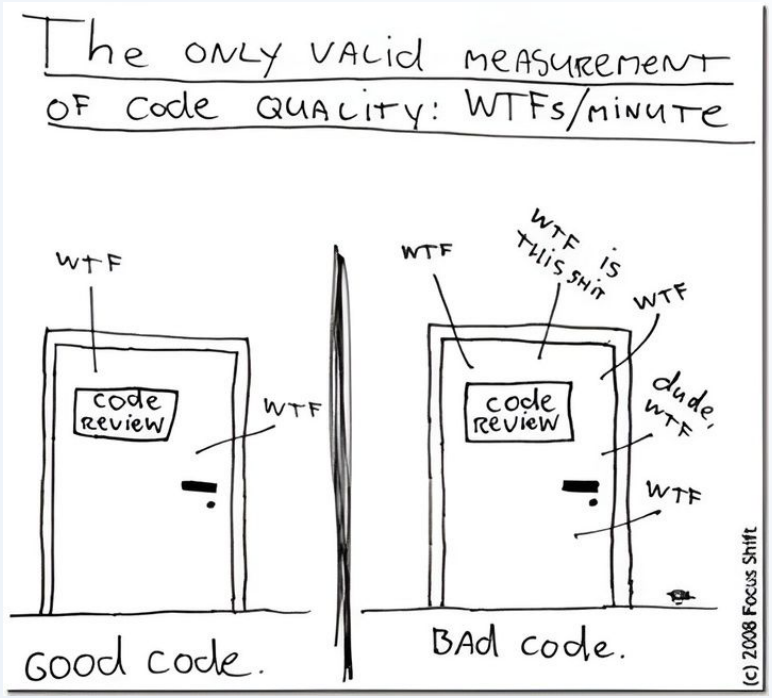
Helen, how can I tell if my code is clean?



Watch the reaction of your reviewers

https://freesvg.org/troll-face

# Measure Clean Code?




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

# Measure Clean Code?

Tools

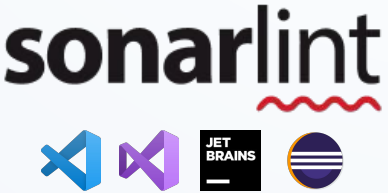
Example project

# Tools

FREE
Community


FREE
public projects


FREE



# Tools

Developers write **clean** code

Teams have a **common** standard


# Who is using Roslyn analyzers?

# Sort(x => x.Downloads)


xUnit.Analyzers - 314M

StyleCop.Analyzers - 108M

Microsoft.Azure.Functions.Analyzers - 31M

Microsoft.VisualStudio.Threading.Analyzers - 30M

SonarAnalyzer.CSharp - 29M

Microsoft.CodeAnalysis.NetAnalyzers - 21M

# Sort(x => x.Downloads)


xUnit.Analyzers

StyleCop.Analyzers - coding style

Microsoft.Azure.Functions.Analyzers

Microsoft.VisualStudio.Threading.Analyzers

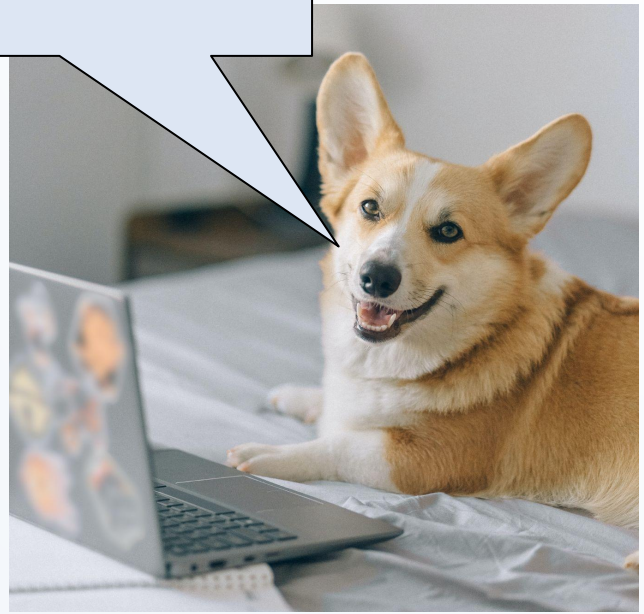
❤️ SonarAnalyzer.CSharp ❤️ sonar

Microsoft.CodeAnalysis.NetAnalyzers - built in


❤️ SonarAnalyzer.CSharp ❤️

Helen, how do tools find problems in our code?

They use static code analysis.



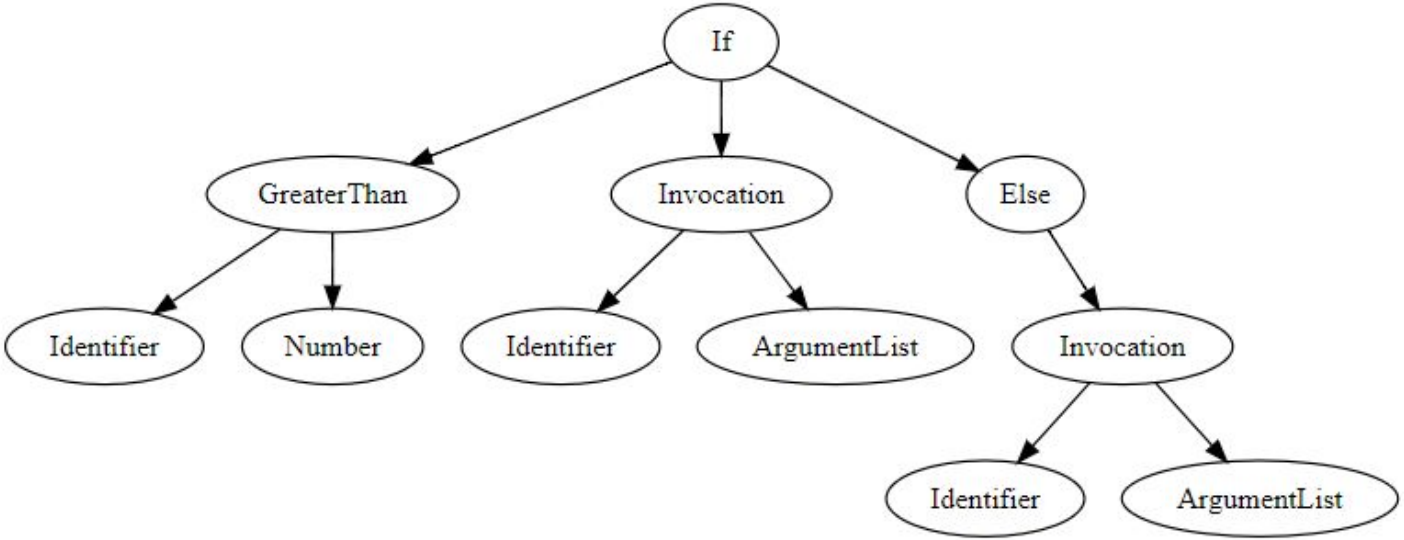
# Static Analysis


Compiler framework

APIs for analyzing code **without** executing it

# Static Analysis

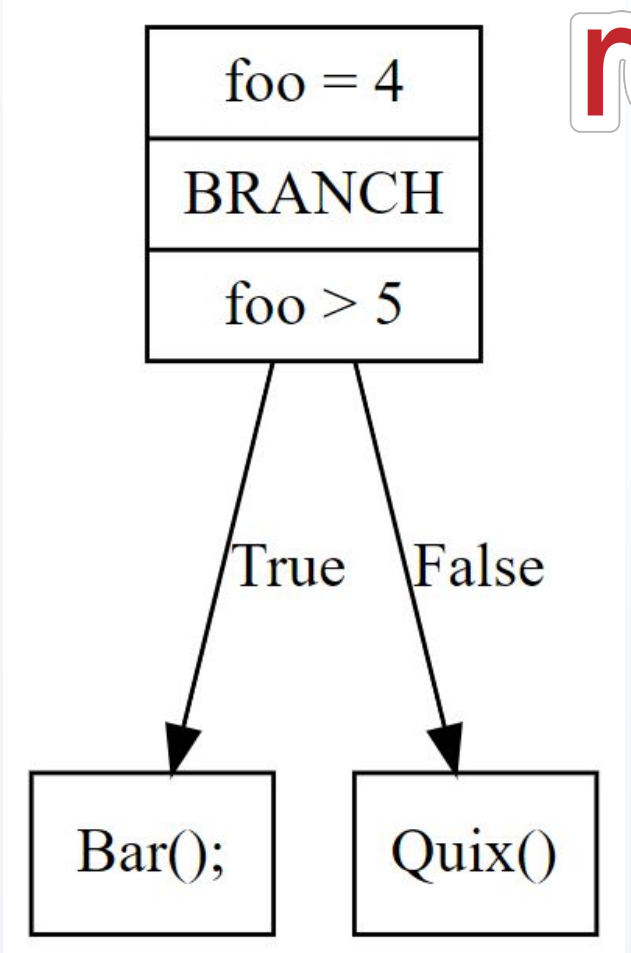
```
if (foo > 5)
    Bar();
else
    Quix();
```




https://edotor.net/

# Static Analysis

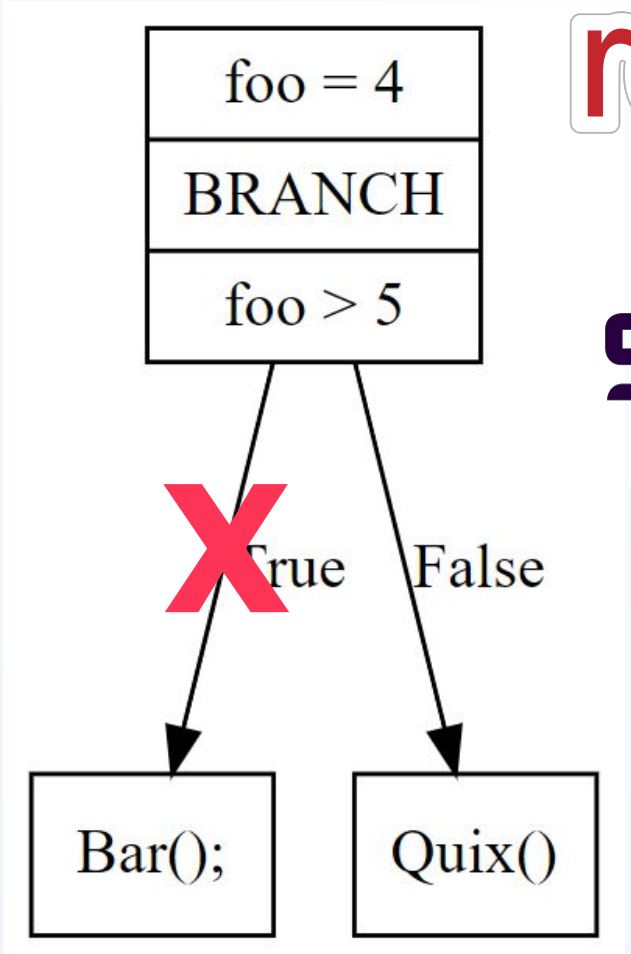
```
var foo = 4;
if (foo > 5)
    Bar();
else
    Quix();
```




roslyn

# Symbolic Execution

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



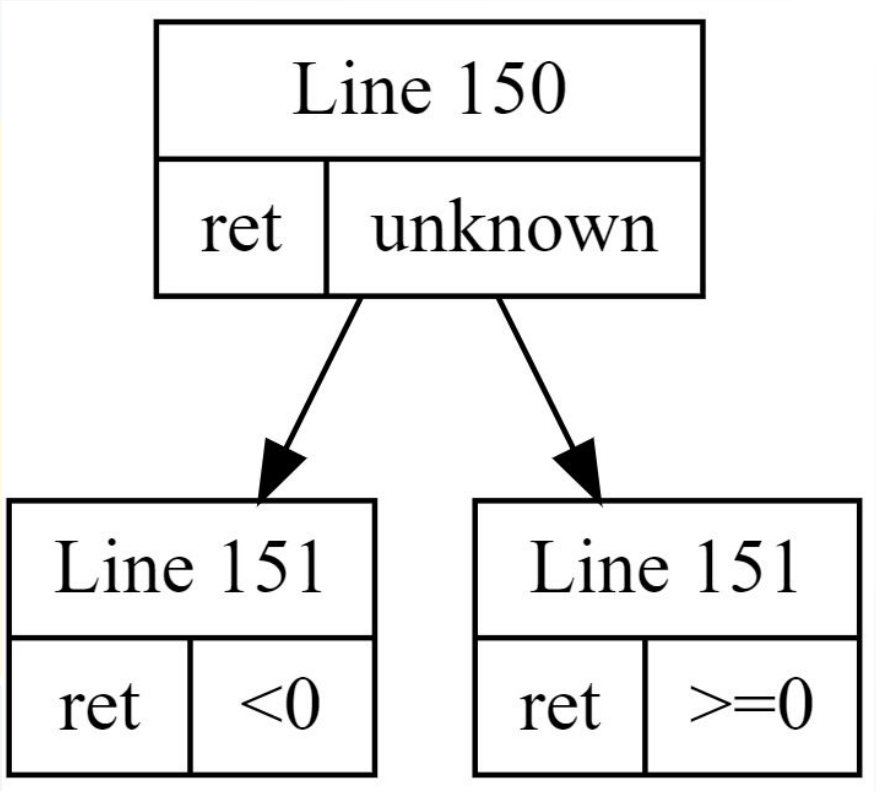
# Symbolic Execution Example

```
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         }
```

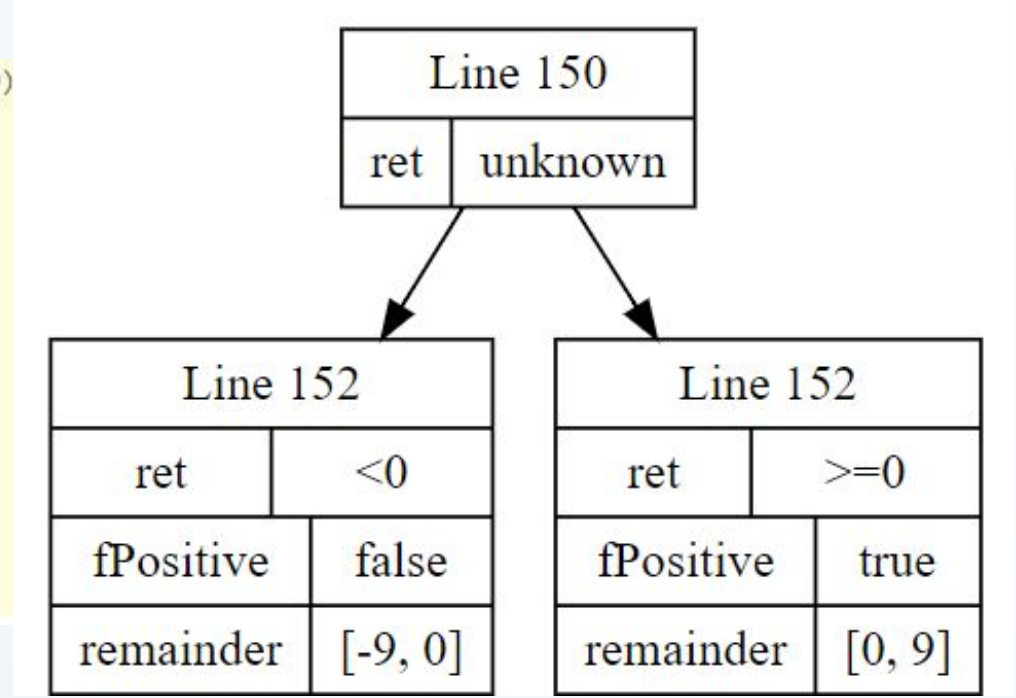
SQLMoney.cs
(dotnet/runtime)

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

```
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             }
```

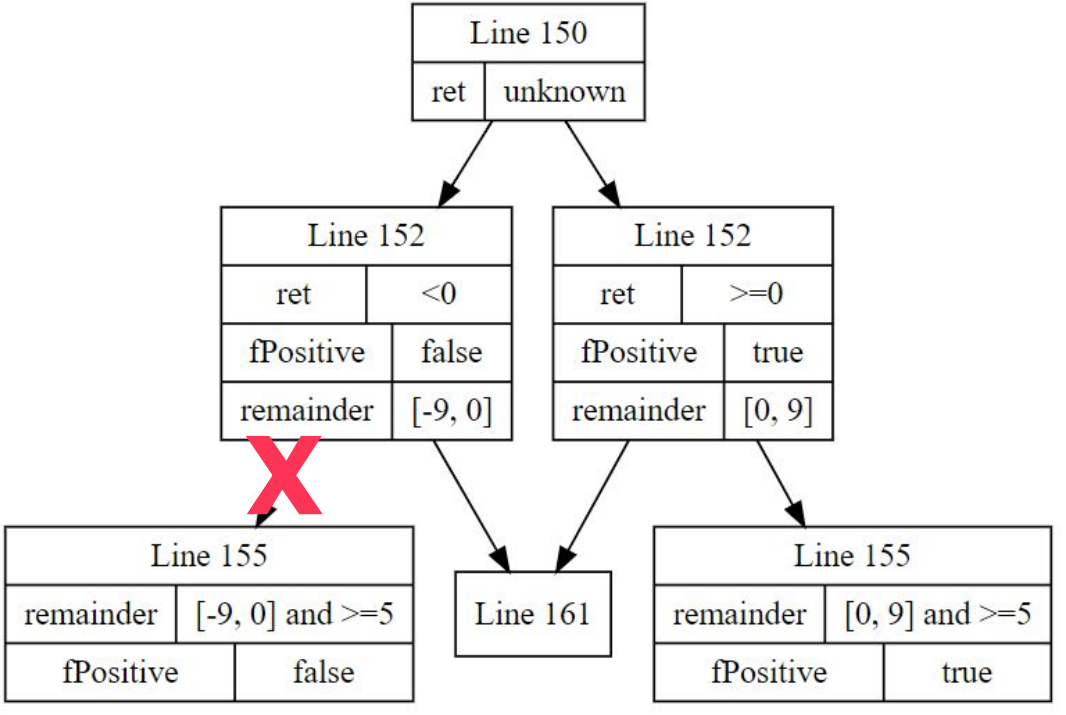


```
150                 long ret = _value / (s_lTickBase / 10);
151                 bool fPositive = (ret >= 0);
152                 long remainder = ret % 10;
153                 ret /= 10;
154
155                 if (remainder >= 5)
156                 {
157                     if (fPositive)
158 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




sharplab

```
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                 }
```



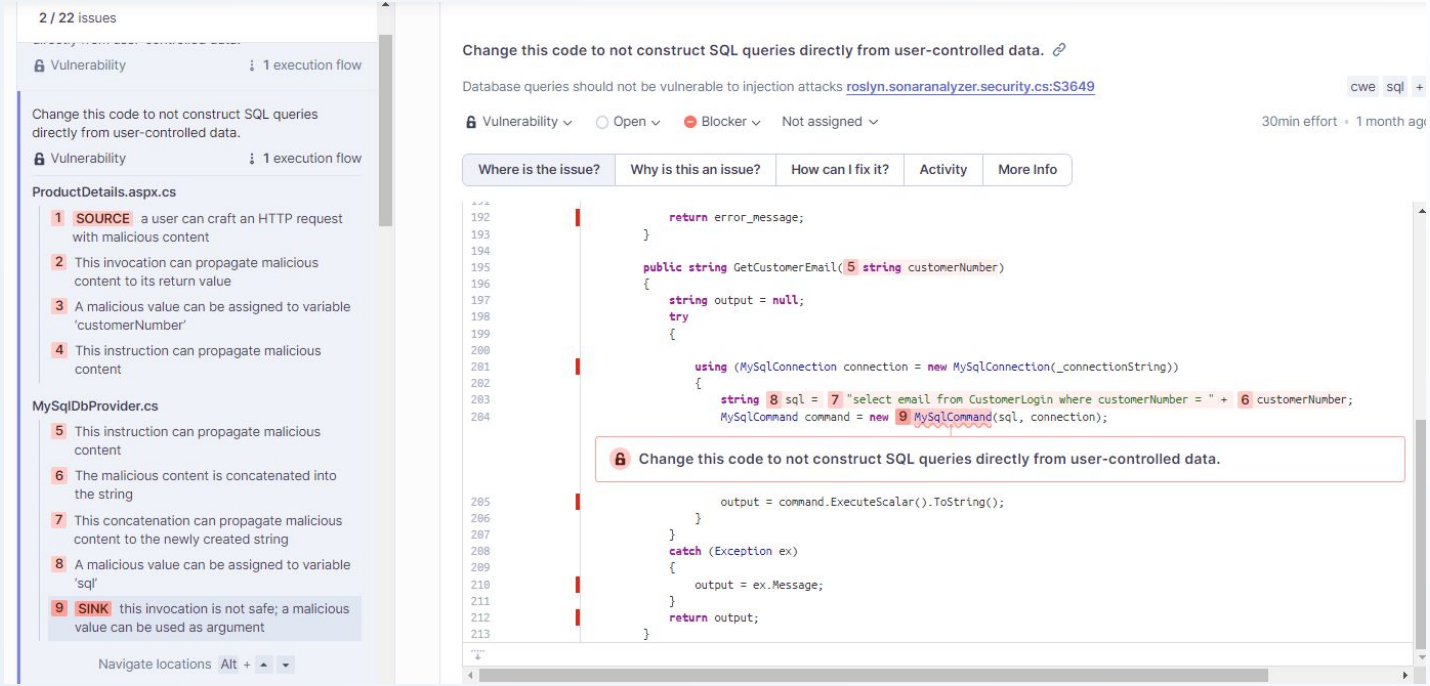
```
150             long ret = _value / (s_lTickBase / 10);
151             bool fPositive = (ret >= 0);
152             long remainder = ret % 10;
153             ret = ret / 10;
154
155             if (remainder >= 5)
156             {
157                 if (fPositive)
```

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;
```

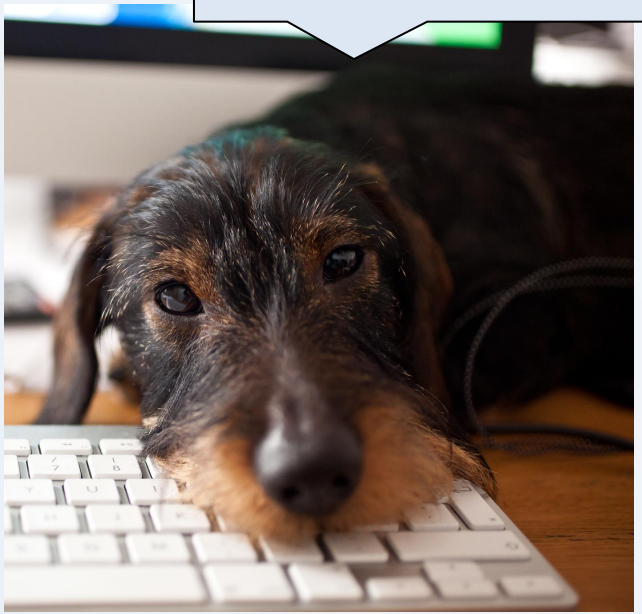
[dotnet/runtime/issues/90741 sharplab]

# Taint analysis

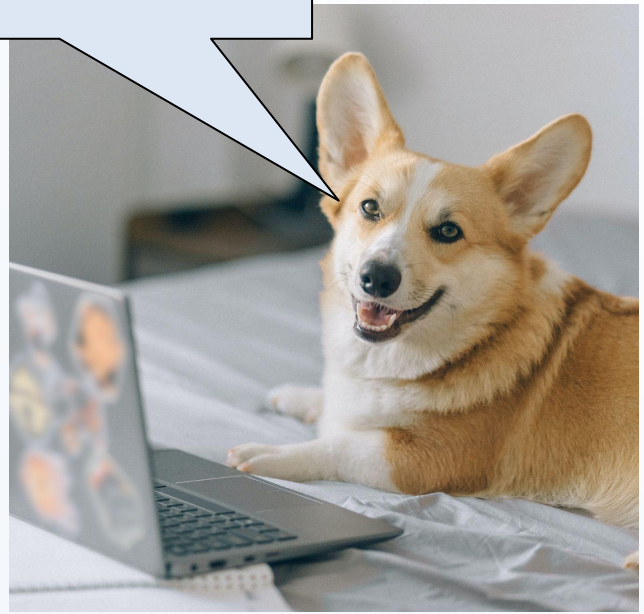



(SonarCloud / SonarQube DE+ only)

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


# 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

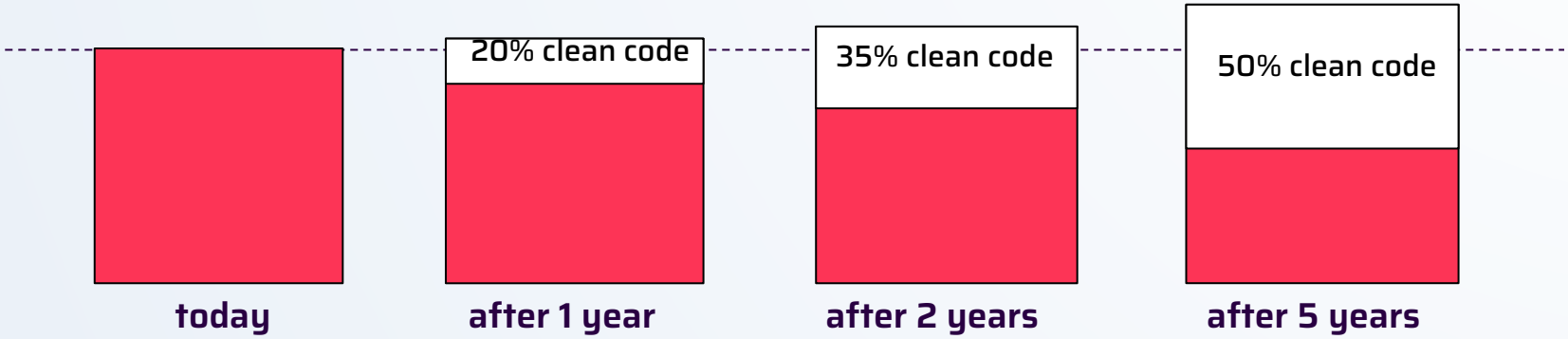
# Clean as You Code

The code is fresh

The cost is ~0

# Clean as You Code

Your existing codebase gets progressively clean



# Implementing Clean as You Code

**New Code Definition**

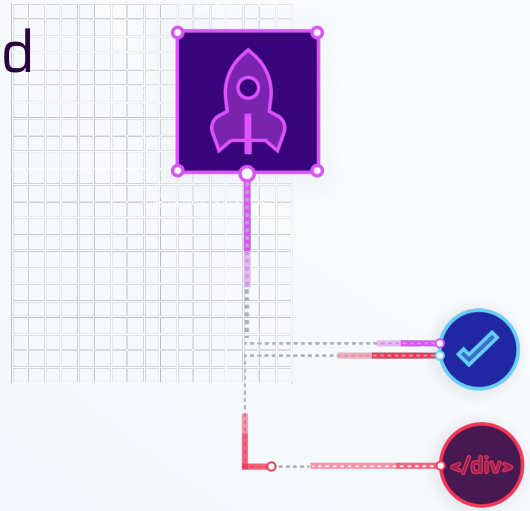
- Pull Request / Commit
- Versions
- Number of days

# 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



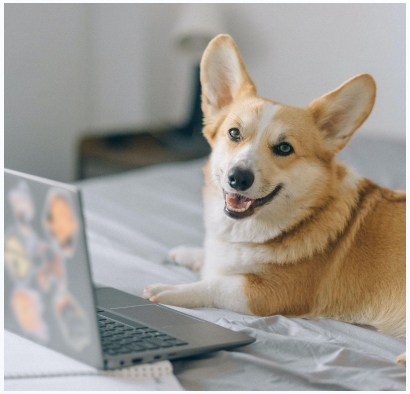
# Clean as You Code

DEMO

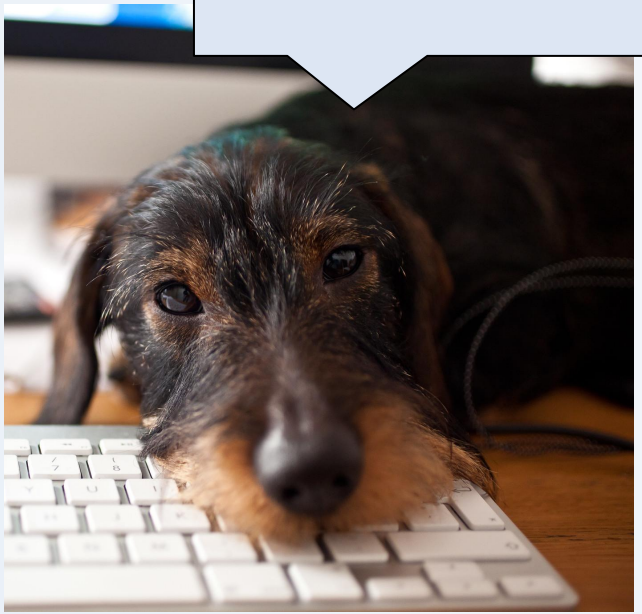
Overall Code vs. New Code

Pull Request integration

SonarLint



I learn as I code



I can focus on more important things during code reviews

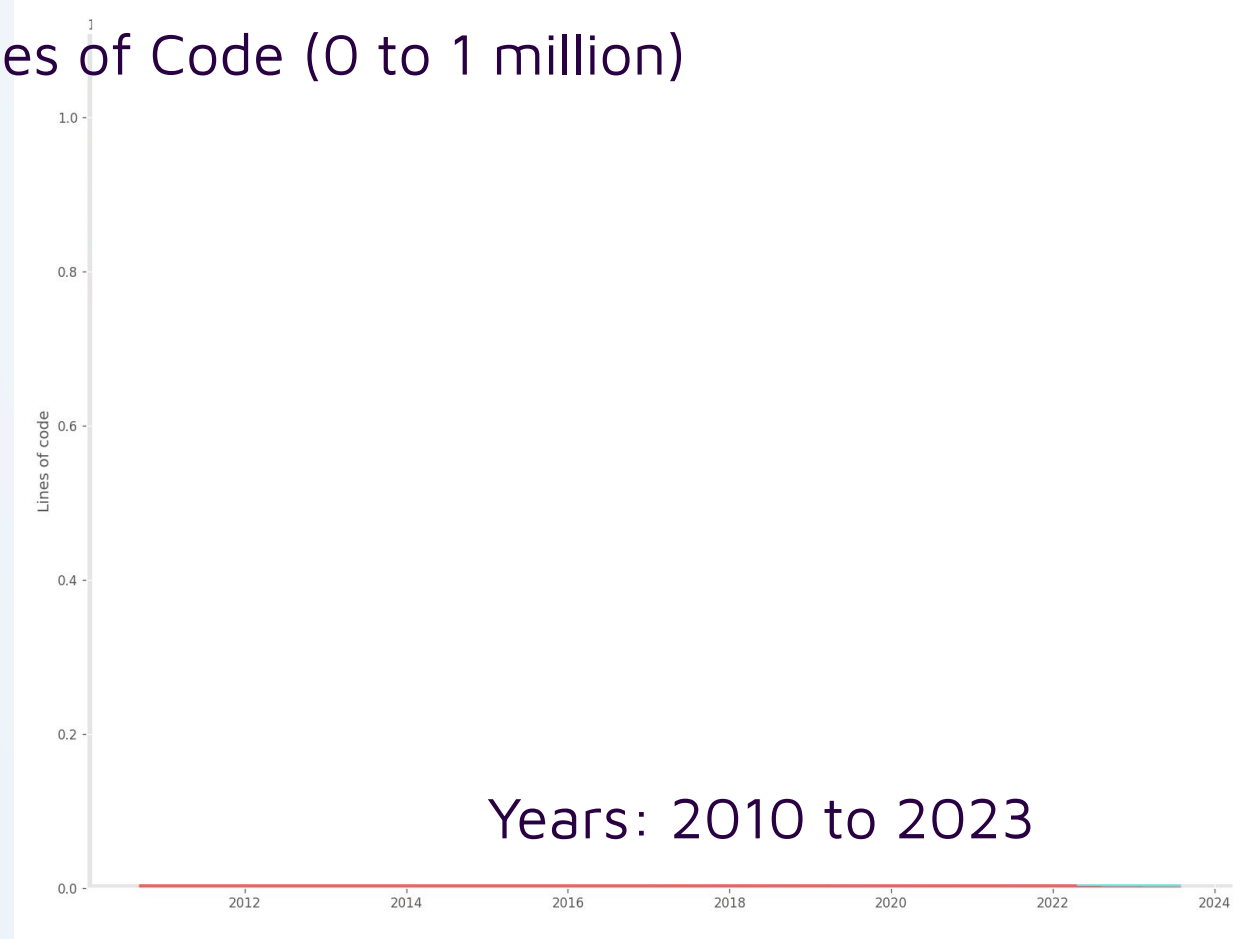
# Why does it work?

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

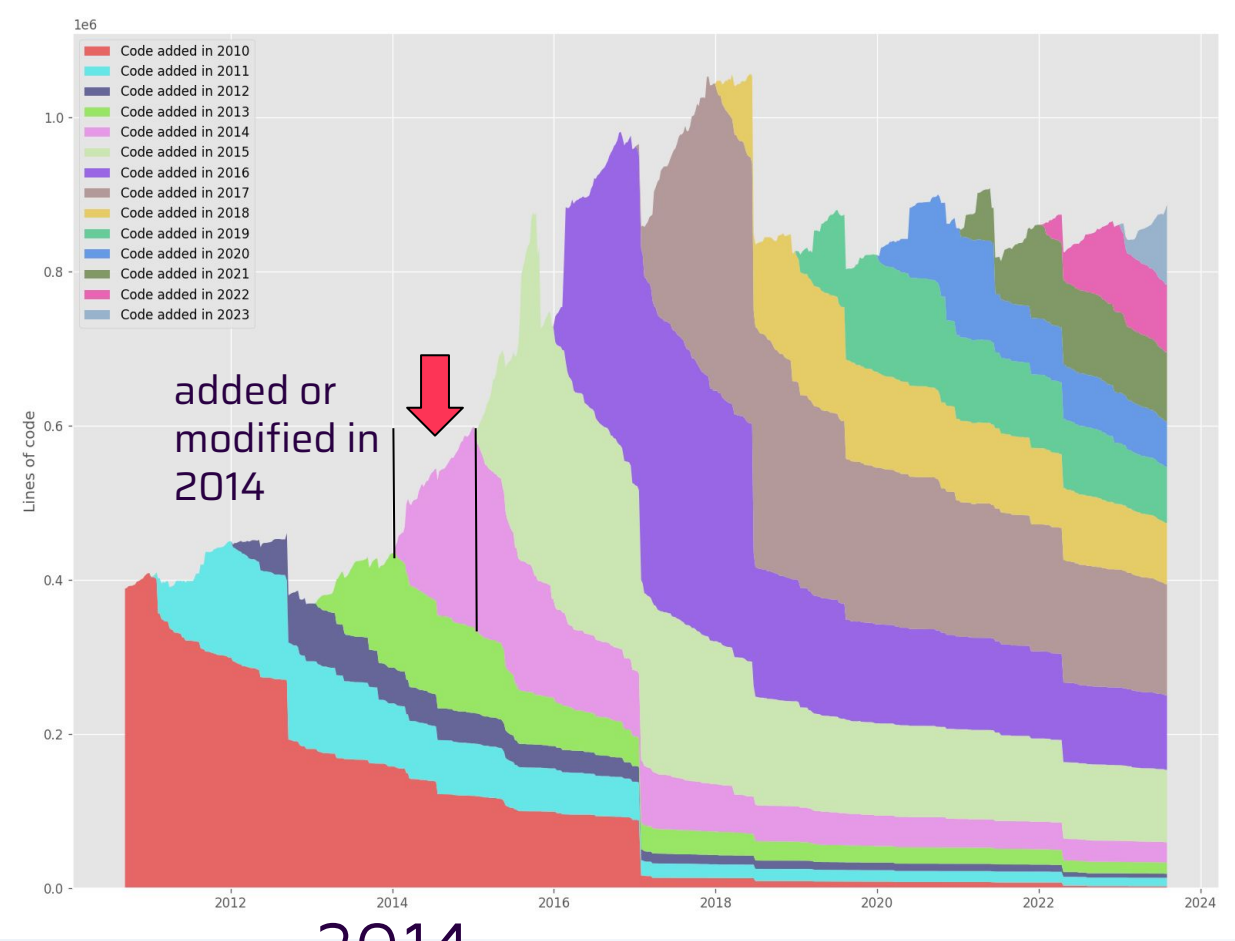
*of large open-source projects on GitHub

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

# Here's SonarQube

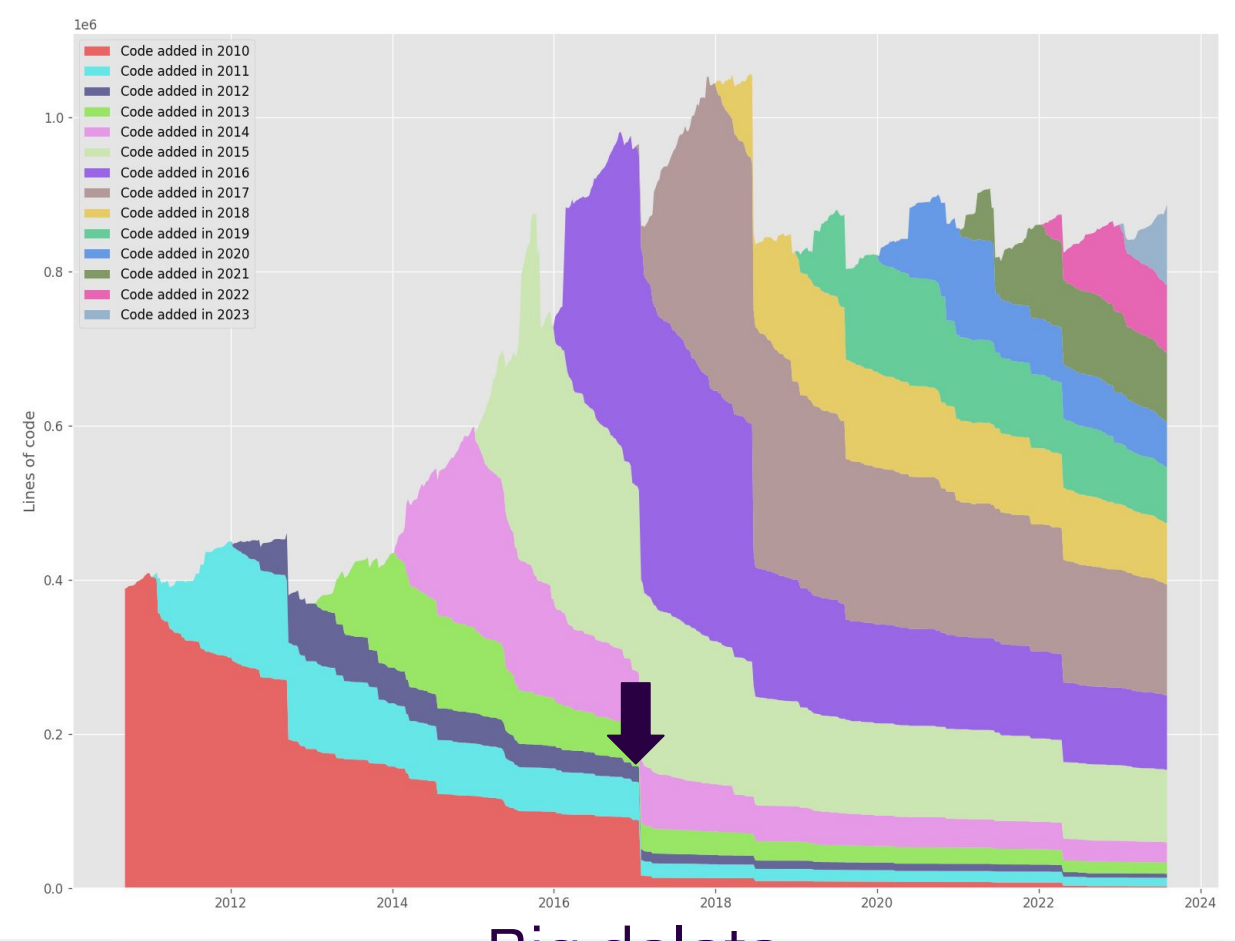


# Here's SonarQube




2014

# Here's SonarQube

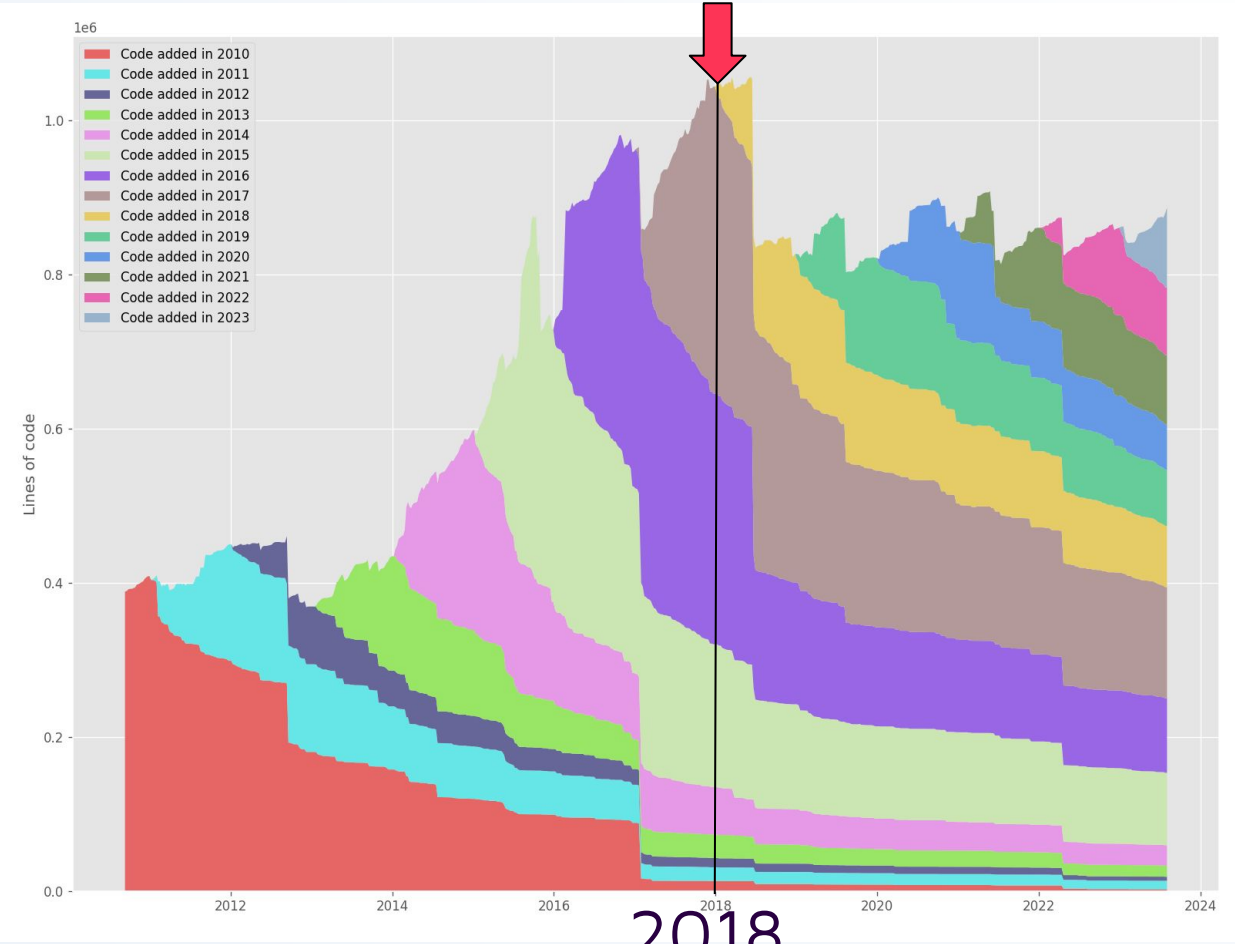



Big delete

(bye-bye ruby code)

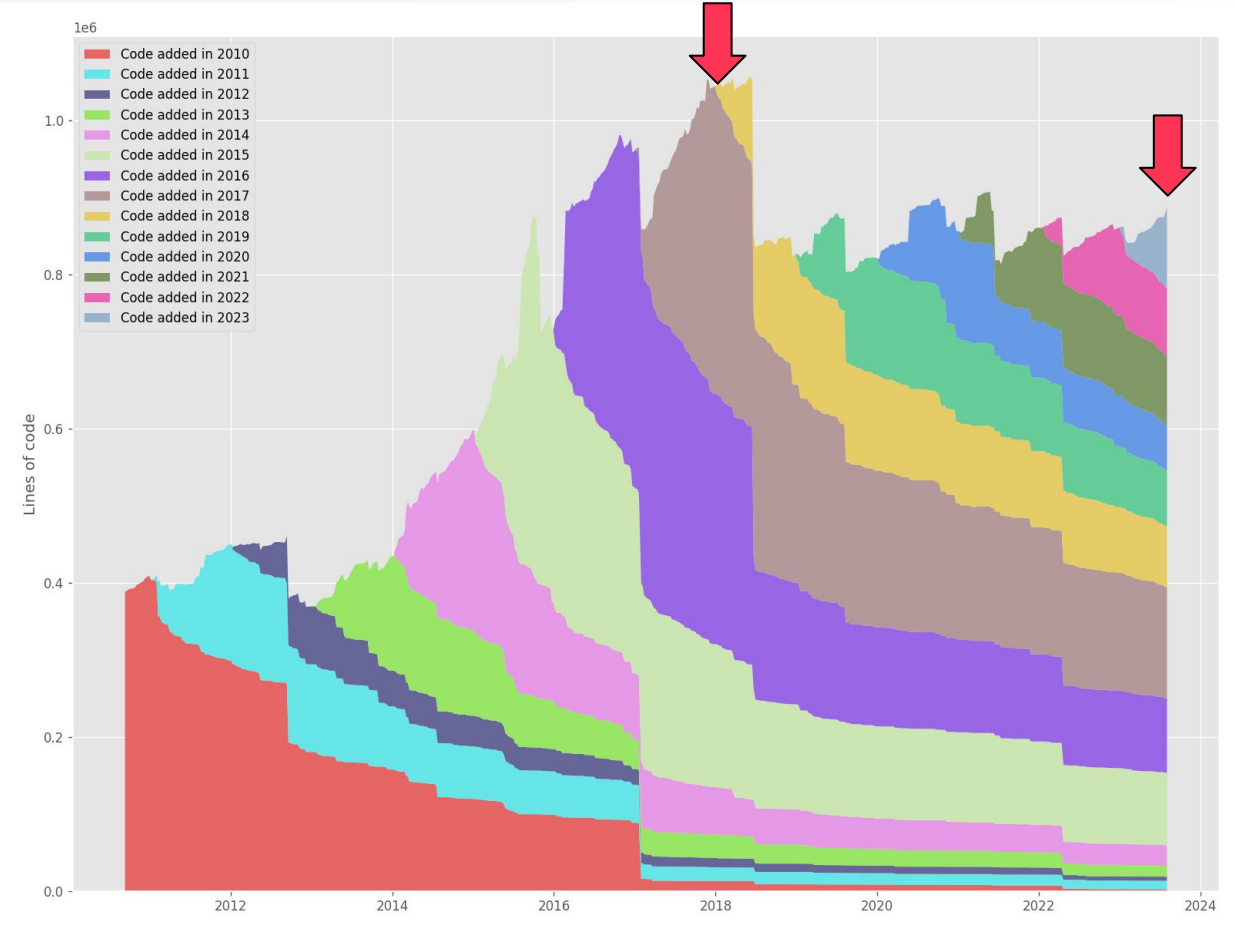
# Here's SonarQube

At the beginning of 2018 there were 1 million LOC



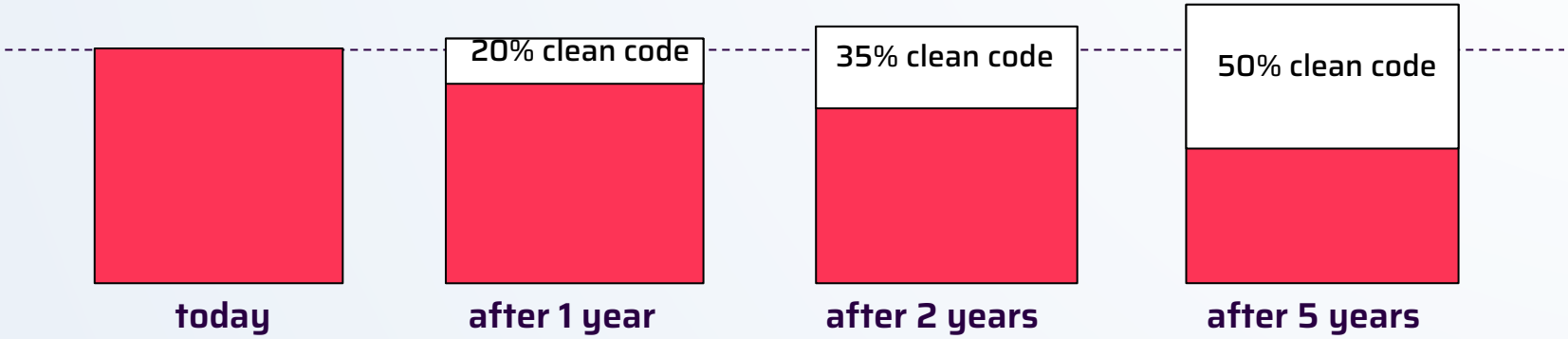
# Here's SonarQube

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

We don’t merge PRs with red QG

Red QG = broken build (slack notification)

# My experience at Sonar

Quality Profile

Quality Gate

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

# 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%.

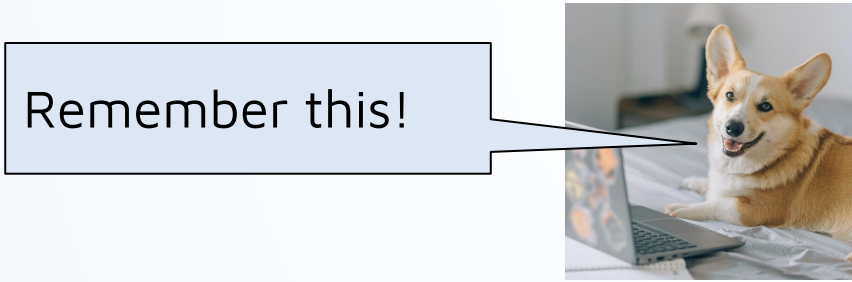
sonar is more

450+ C# Rules

30+ languages, frameworks, infra technologies

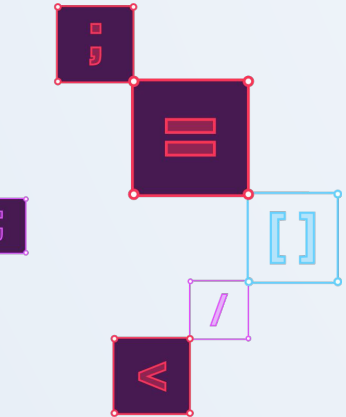
rules.sonarsource.com

# 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



# Feedback form & slides:
# AndreiEpure.ro

# Extra slides

# My experience at Sonar

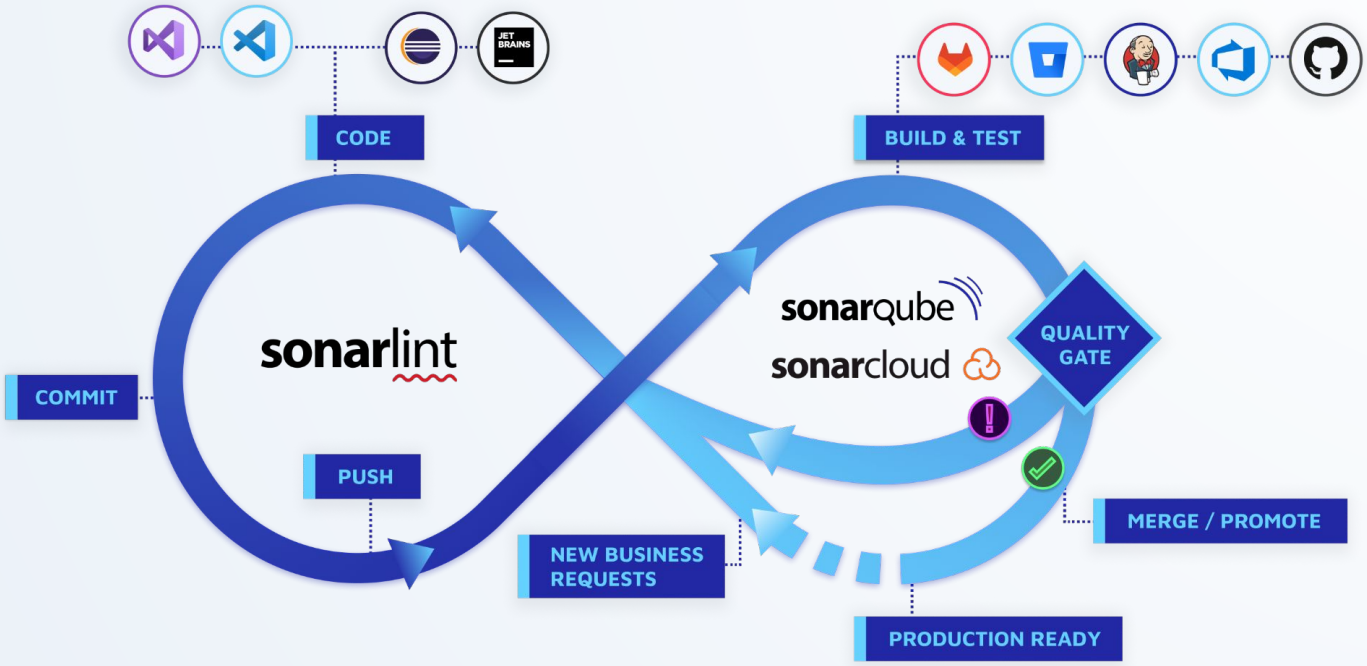
New issues always appear on the “overall code”

(new rules, improved techniques).

# My experience at Sonar

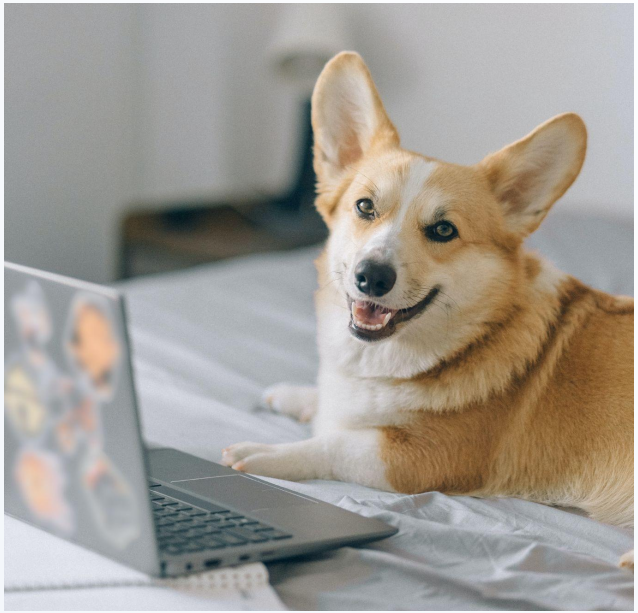
Human code review cannot be replaced.

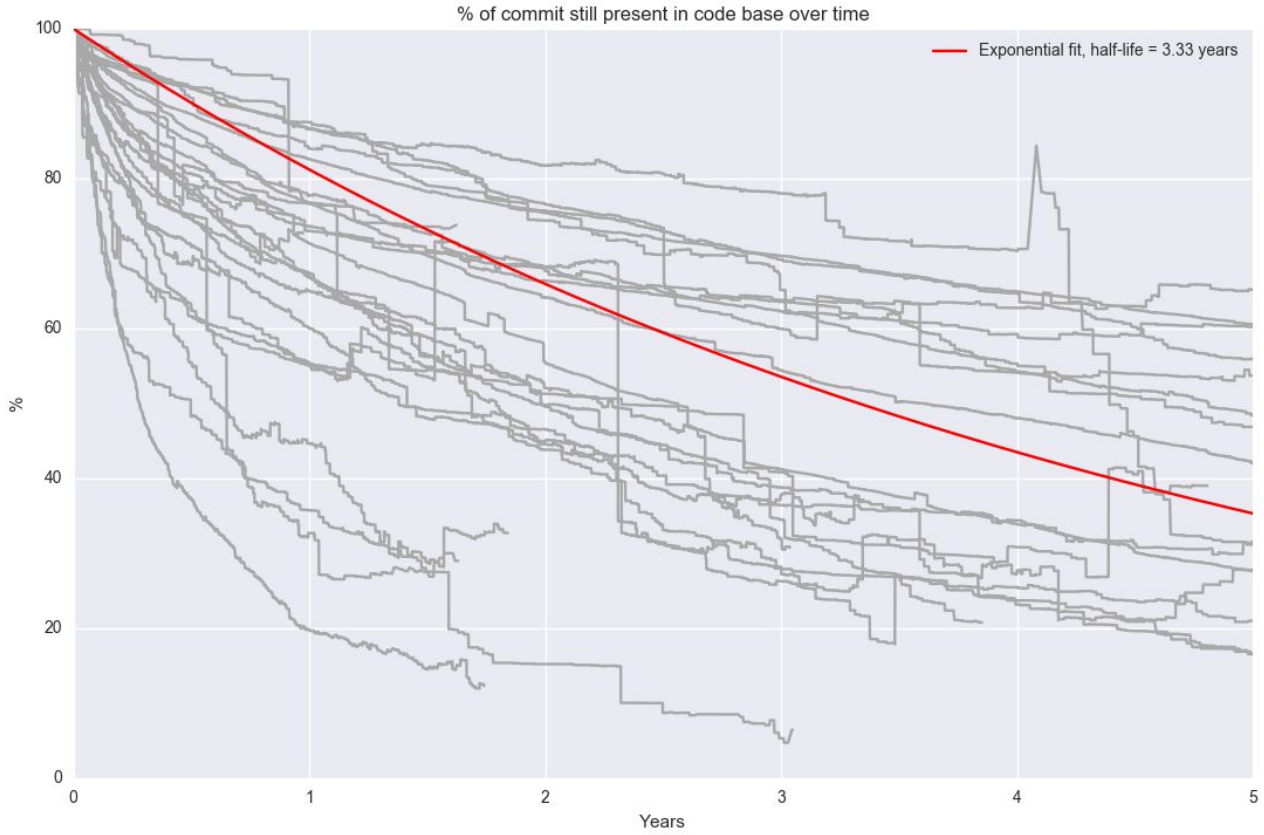
# Part of Development



# Clean as You Code

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





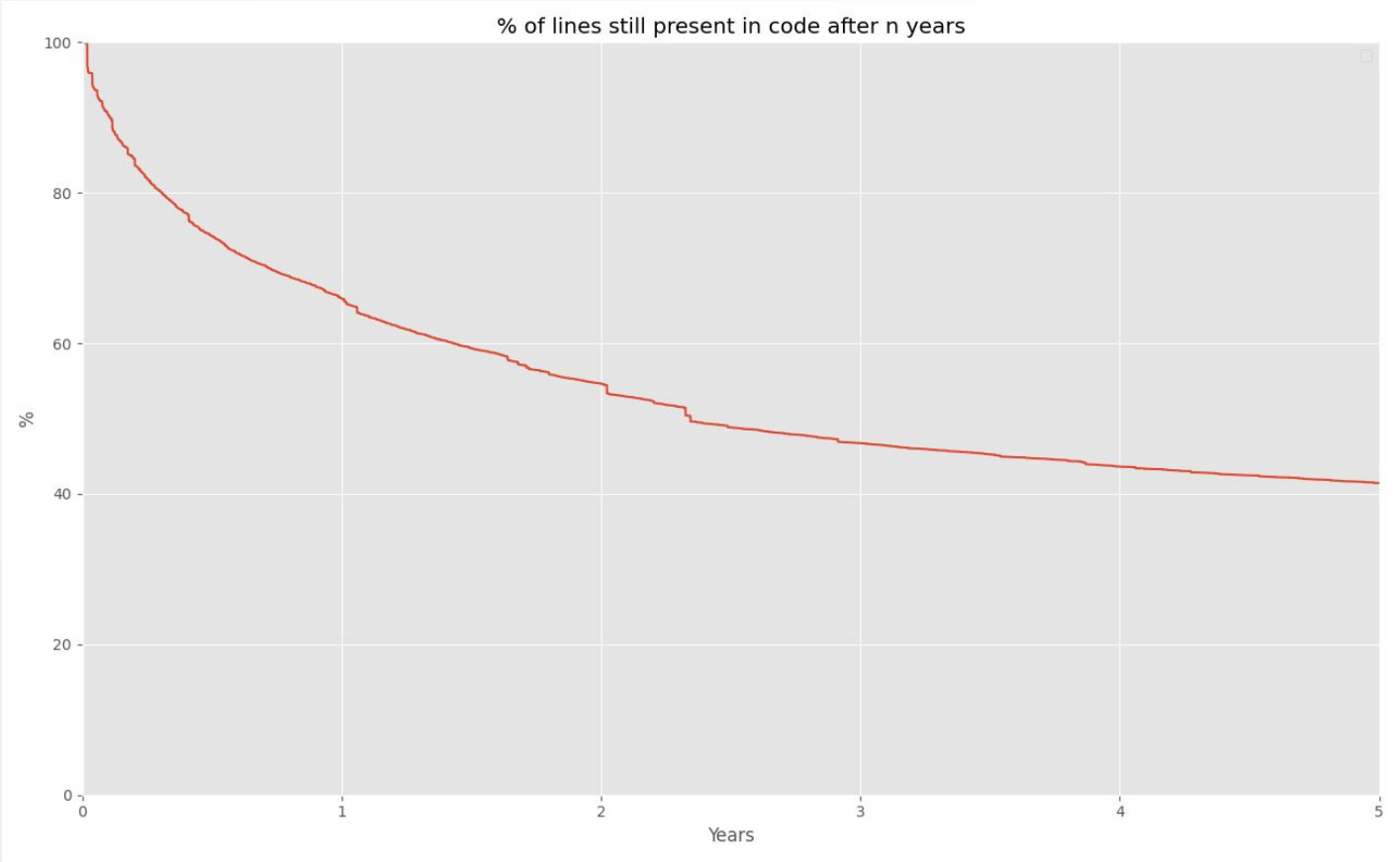





Erik Bernhardsson - “The half-life of code & the ship of Theseus” (2016)

# Here's SonarQube

In 5 years, more than 50% of the code has been changed.



# Software is the fruit of code