



Clean as You Code

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

Andrei Epure

Developer

Engineering Manager at



❤️ 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



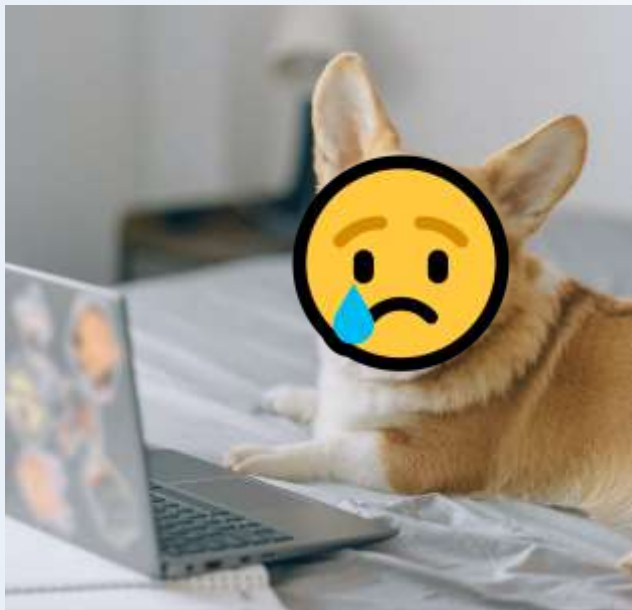
Reliable (no bugs)

Secure (no vulnerabilities)

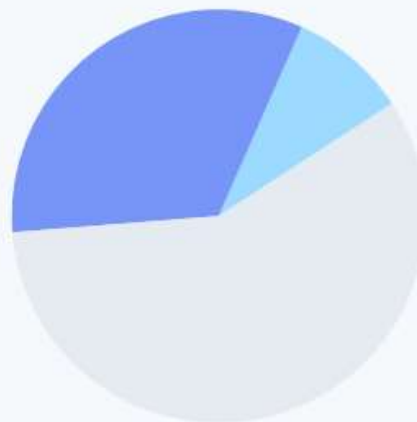
Maintainable (no code smells)

...

Clean Code: development and production



THE DEVELOPER WORK WEEK



- 13.5 hours
Technical debt
- 3.8 hours
Bad code

41.1 total hours

Average developer work week

<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**

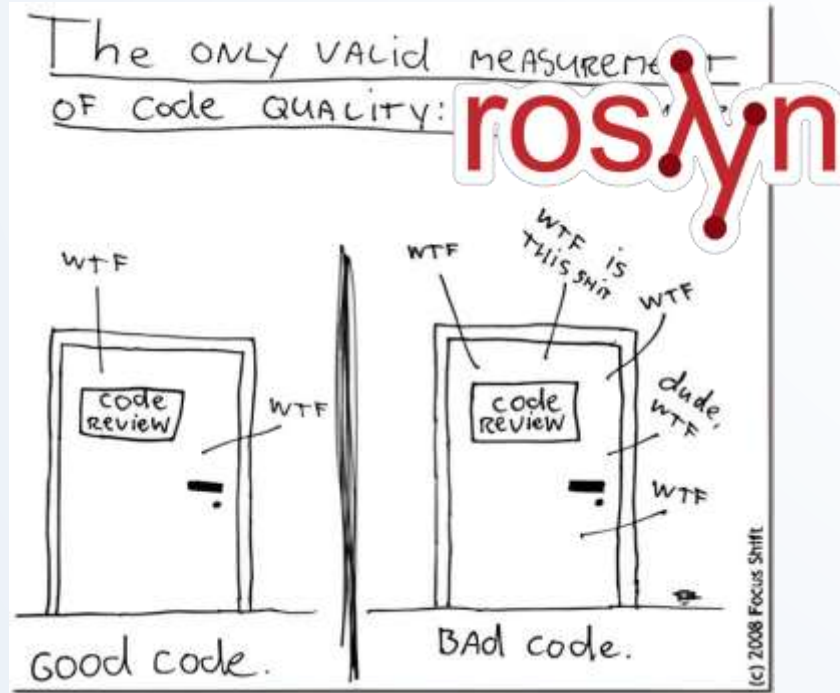


**Helen Today
Standard**



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



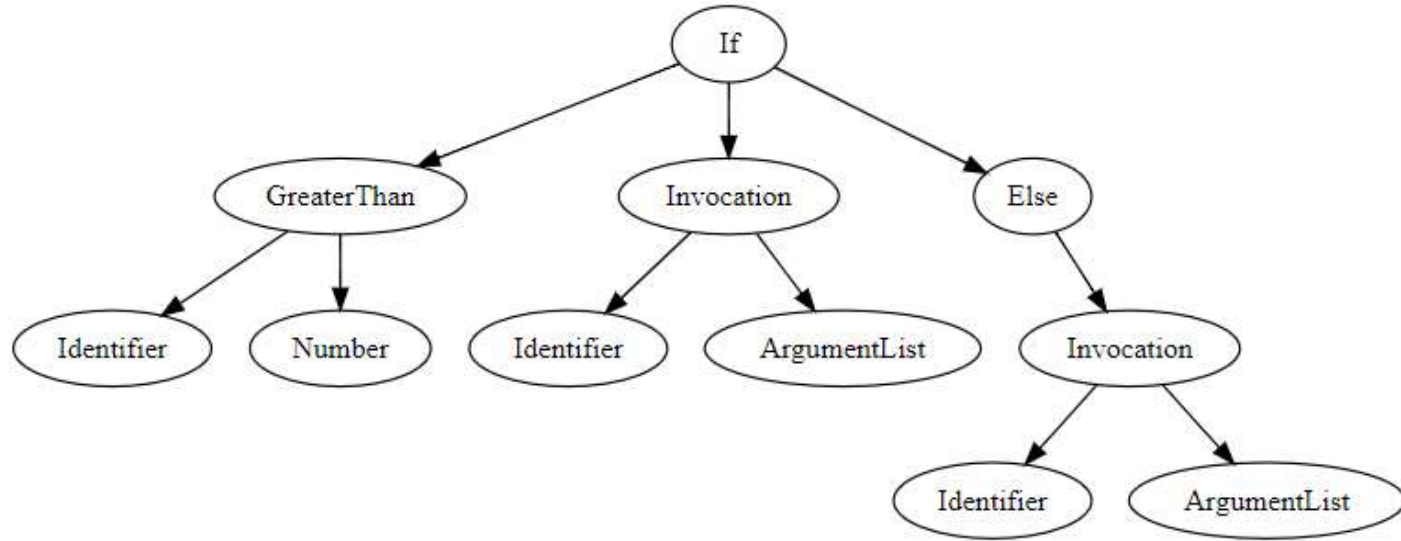
Compiler framework for .NET

APIs for analyzing code **without** executing it

Syntax Tree



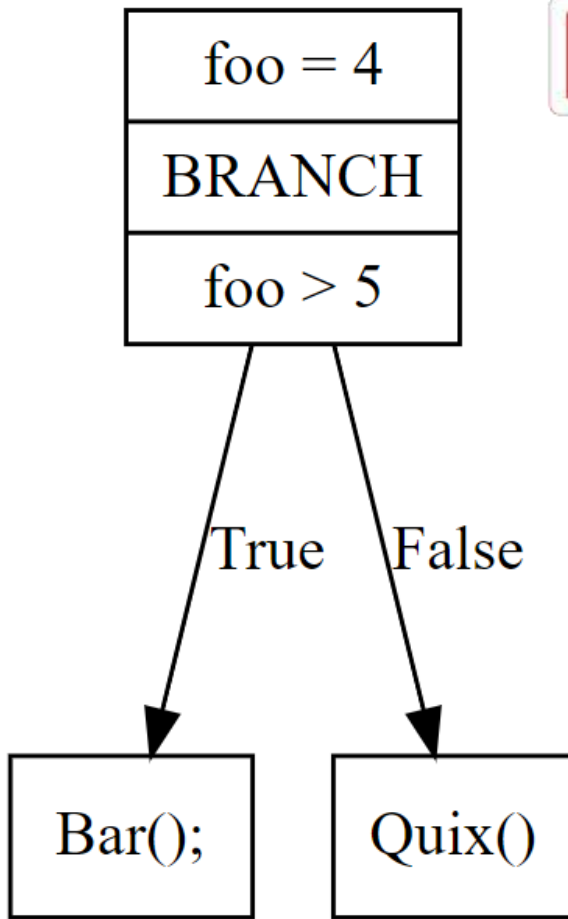
```
if (foo > 5)  
    Bar();  
else  
    Quix();
```



<https://edotor.net/>

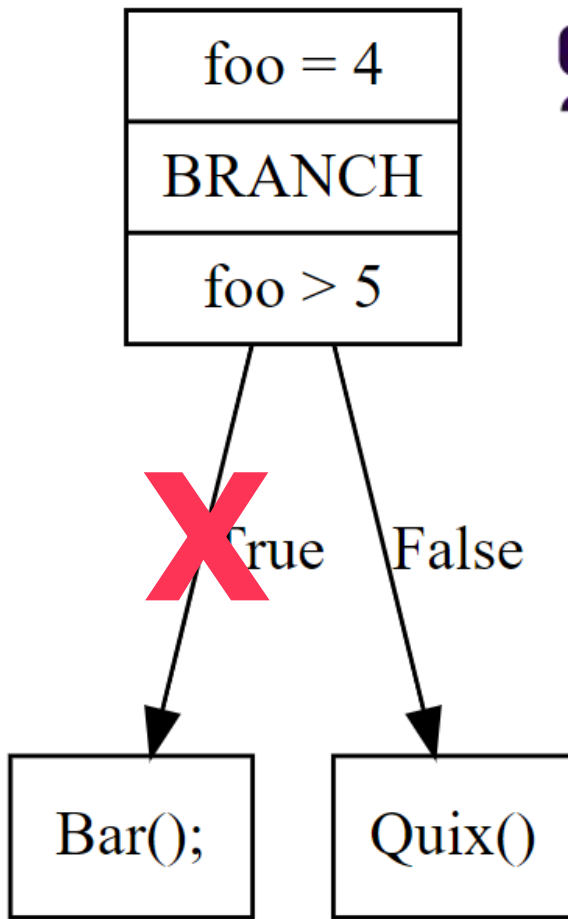
Control Flow Graph

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



Symbolic Execution

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



Symbolic Execution Example

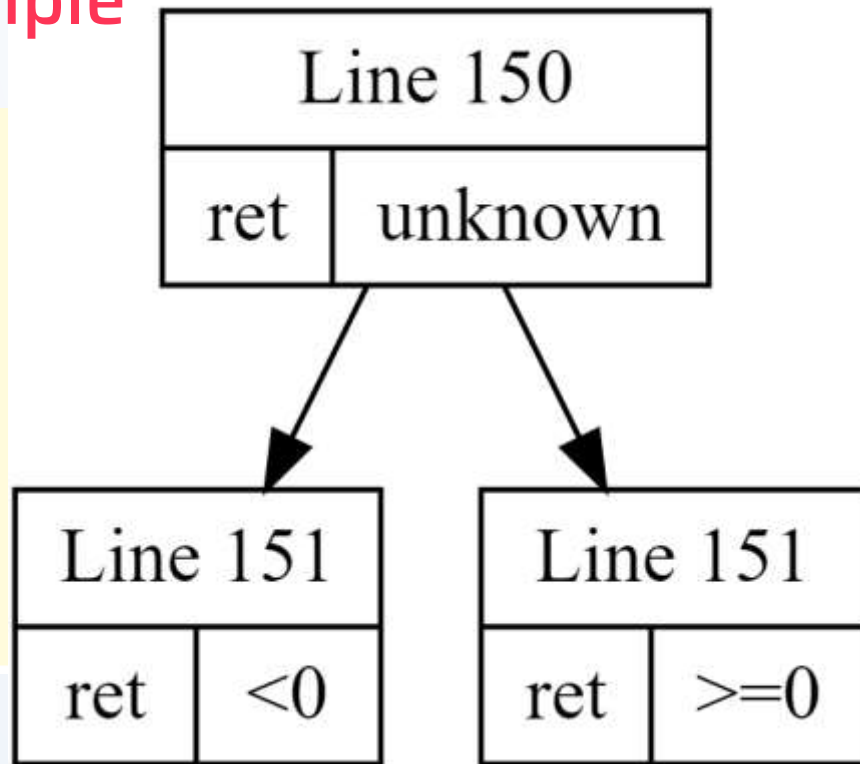
```
145  ✓      public long ToInt64()  
146          {  
147              if (IsNull)  
148                  throw new SqlNullValueException();  
149  
150              long ret = _value / (s_lTickBase / 10);  
151              bool fPositive = (ret >= 0);  
152              long remainder = ret % 10;  
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)

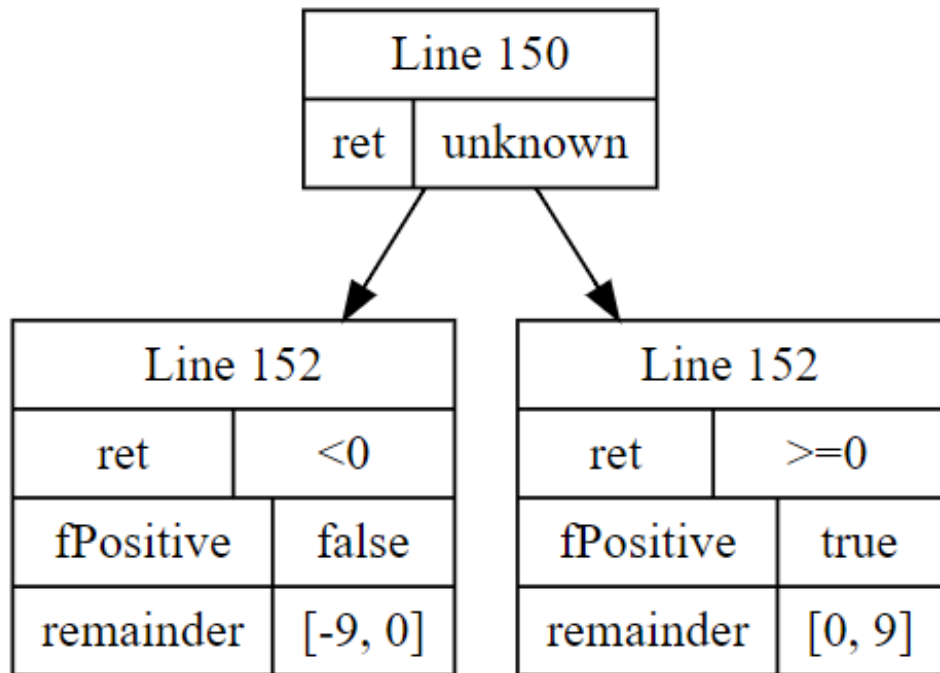
Symbolic Execution Example

```
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      }
```



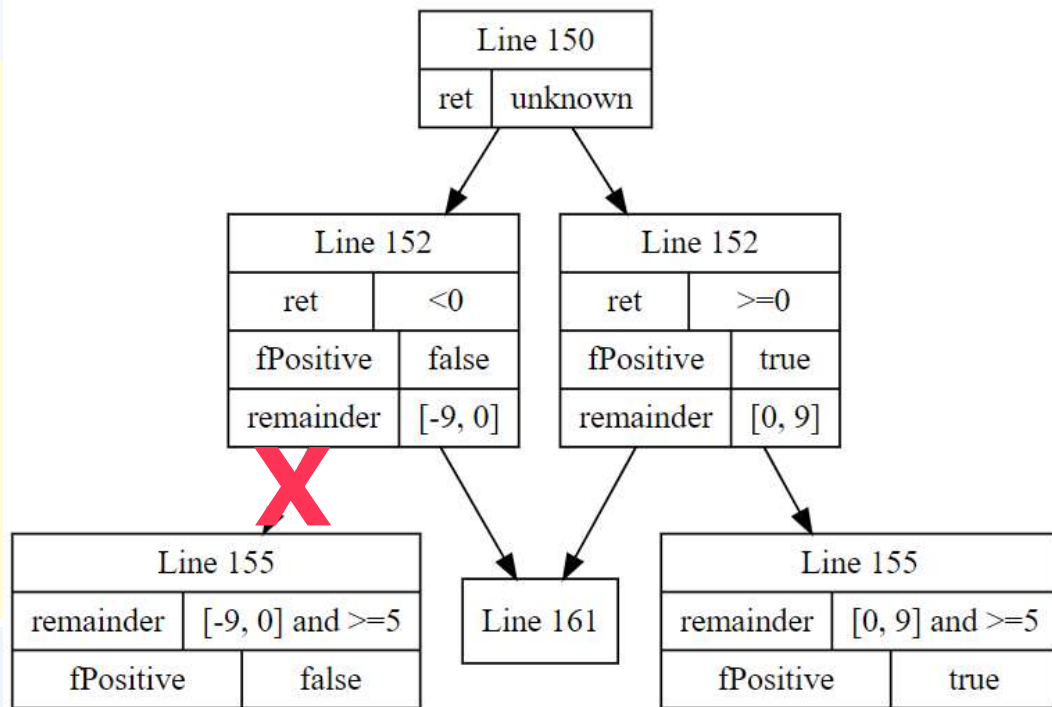
sharplab

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     {
161         ret--;
162     }
163     return ret;
```

 dotnet/runtime/issues/90741

 [sharplab](#)

Taint analysis

2 / 22 Issues

Vulnerability 1 execution flow

Change this code to not construct SQL queries directly from user-controlled data.

Vulnerability 1 execution flow

ProductDetails.aspx.cs

- SOURCE** a user can craft an HTTP request with malicious content
- This invocation can propagate malicious content to its return value
- A malicious value can be assigned to variable 'customerNumber'
- This instruction can propagate malicious content

MySqlDbProvider.cs

- This instruction can propagate malicious content
- The malicious content is concatenated into the string
- This concatenation can propagate malicious content to the newly created string
- A malicious value can be assigned to variable 'sql'
- SINK** this invocation is not safe; a malicious value can be used as argument

Navigate locations: All + -

Change this code to not construct SQL queries directly from user-controlled data. [roslyn.sonaranalyzer.security.cs:53640](#)

Database queries should not be vulnerable to injection attacks

Vulnerability = ☐ Open ☒ Blocker ☐ Not assigned

30min effort - 1 month ago

Where is the issue?	Why is this an issue?	How can I fix it?	Activity	More info
<pre> 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 </pre>	<p>Change this code to not construct SQL queries directly from user-controlled data.</p>	<pre> 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 </pre>		

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



Who is using Roslyn analyzers?

Setup analyzers

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

Configure rules with **.editorconfig**

Demo



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

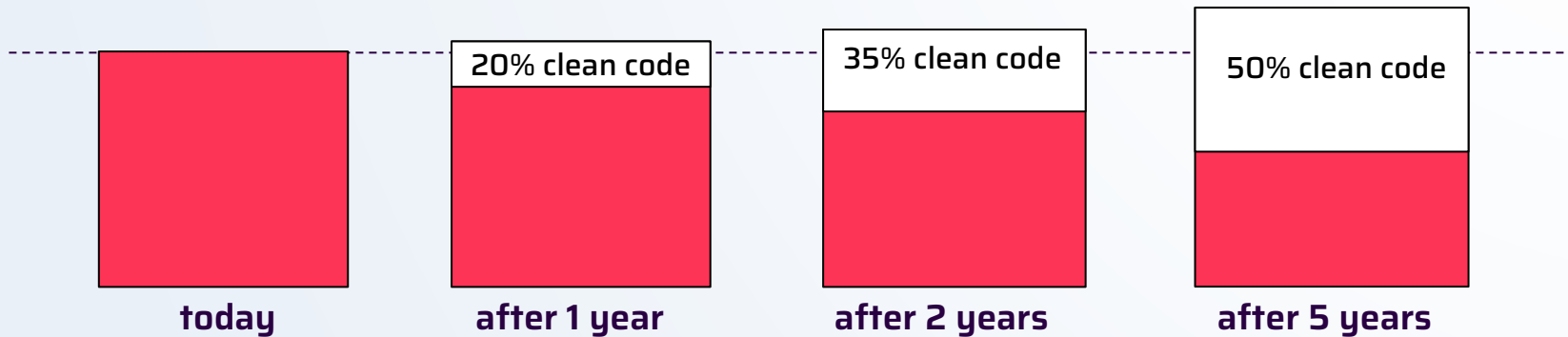
Clean as You Code

The code is fresh

The cost is ~0

Clean as You Code

Your existing codebase gets progressively clean

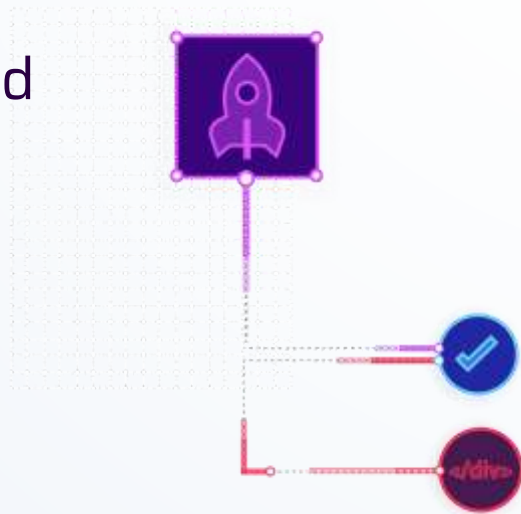


Implementing Clean as You Code

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)

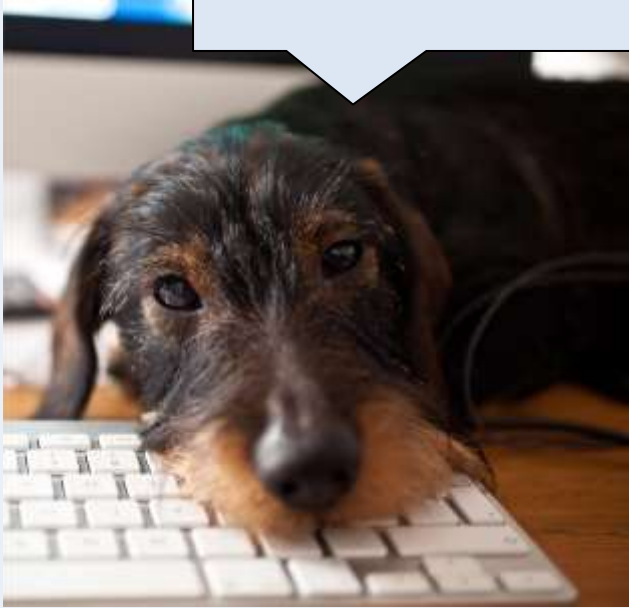
Overall Code vs. New Code

👉 Pull Request integration

SonarLint



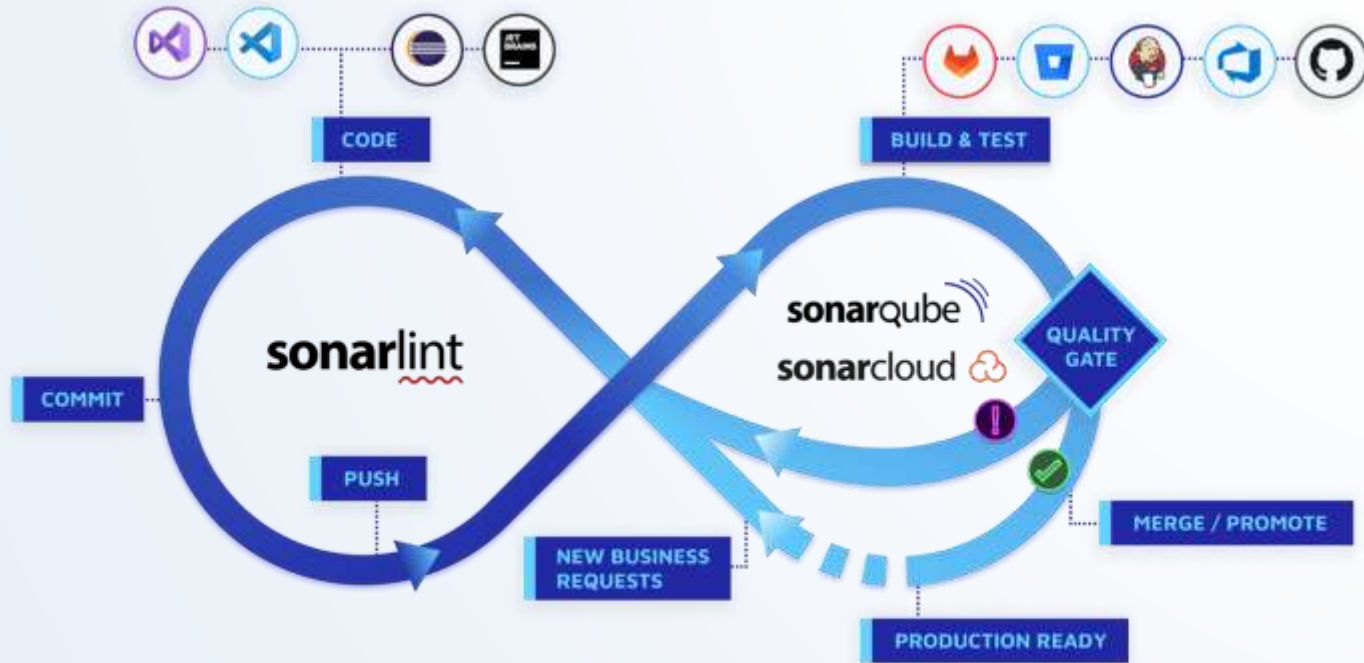
I learn as I code



I can focus on more important things during code reviews



Part of Development



Why does it work?

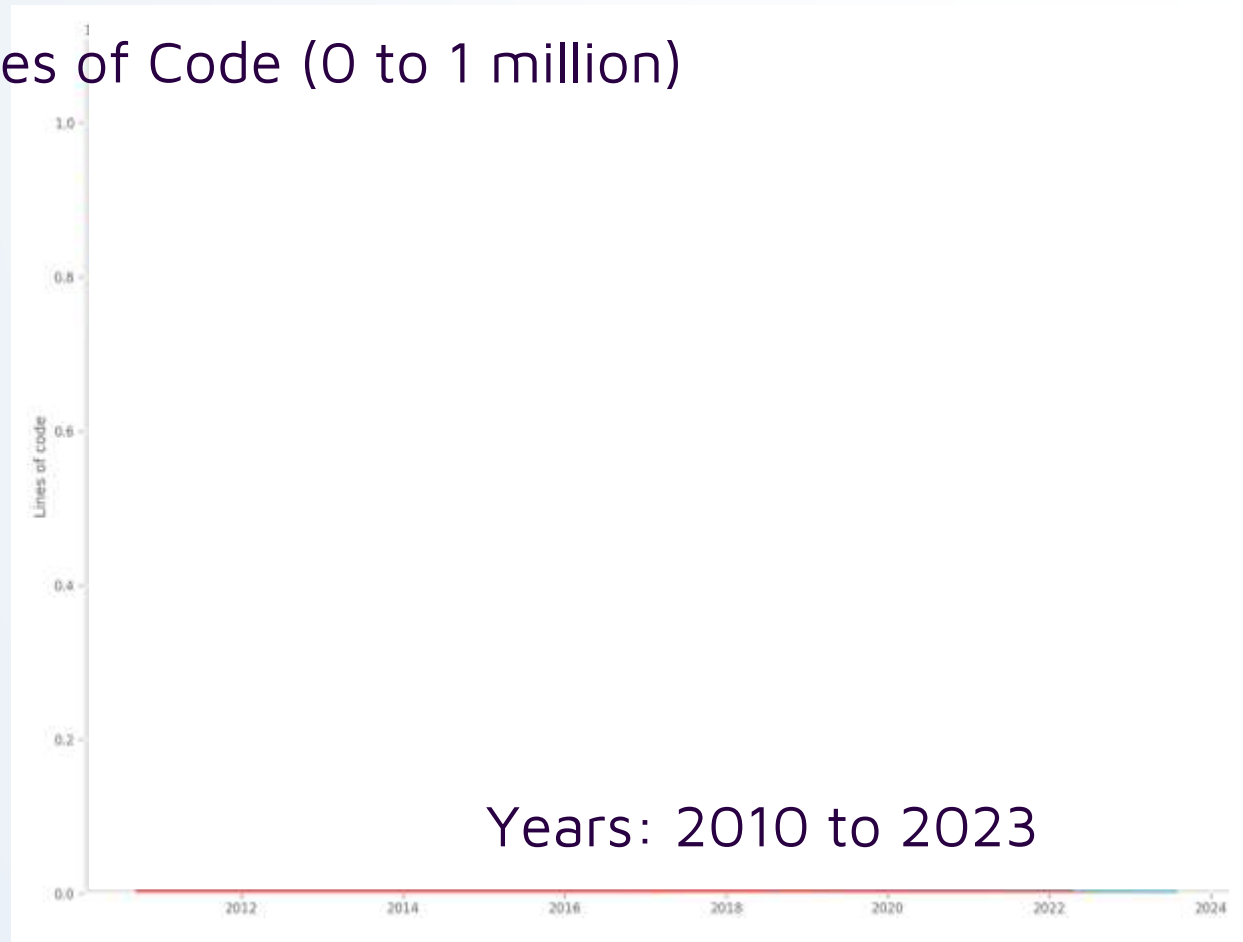
On average, 50% of code* gets changed within 3.33 years.

*of large open-source projects on GitHub

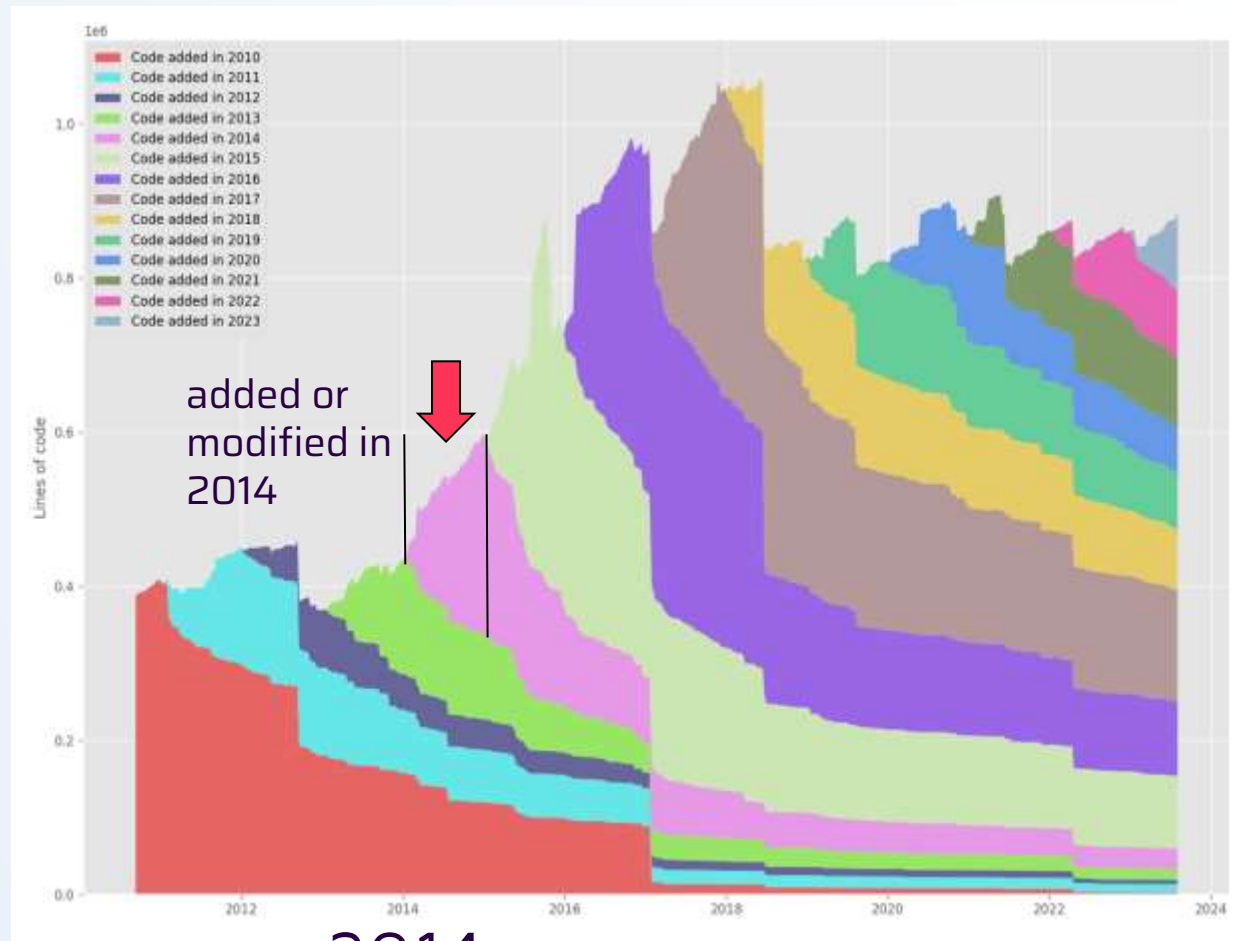
👉 <https://github.com/erikbern/git-of-theseus>

Here's SonarQube

Lines of Code (0 to 1 million)



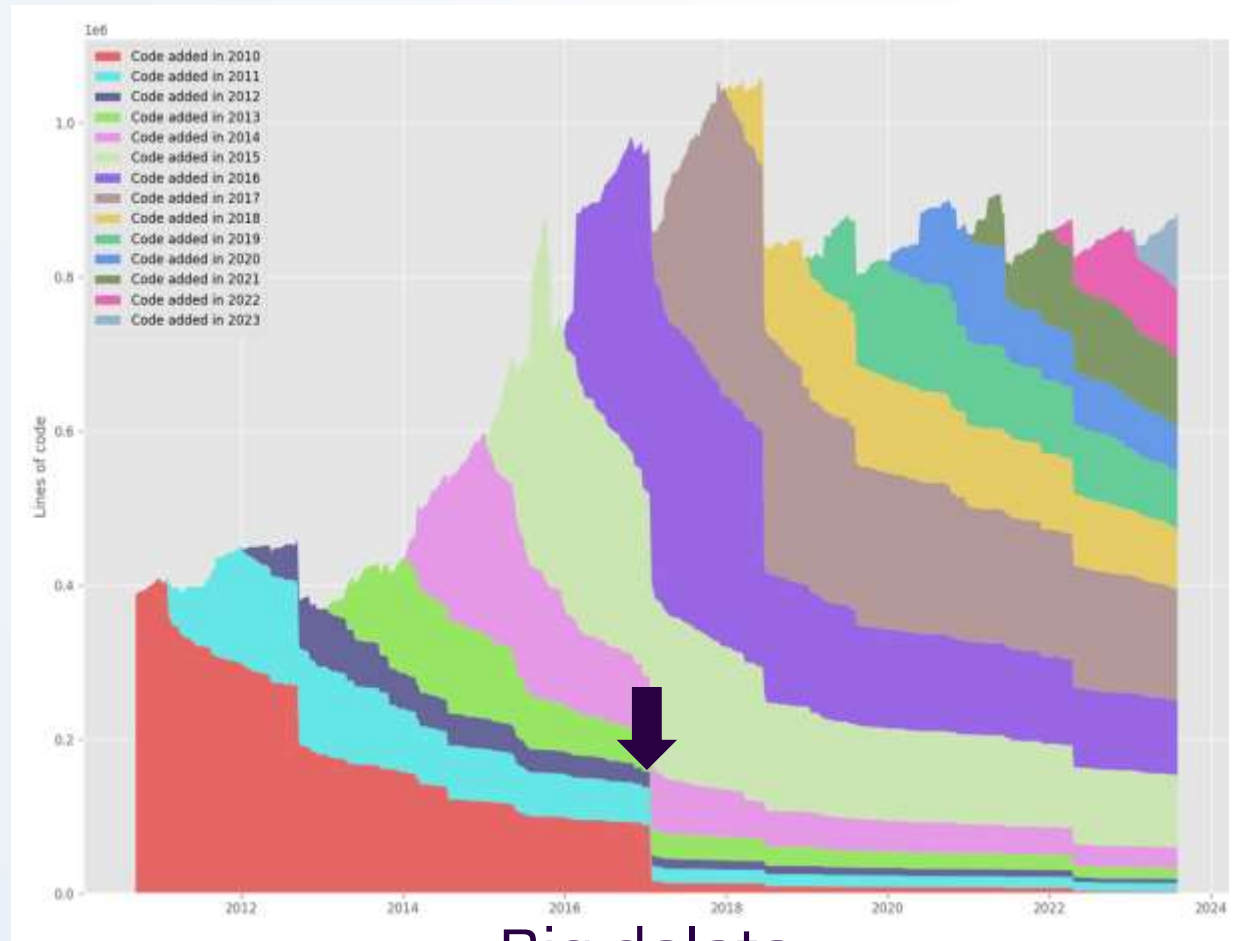
Here's SonarQube



2014

©2023, SonarSource S.A, Switzerland.

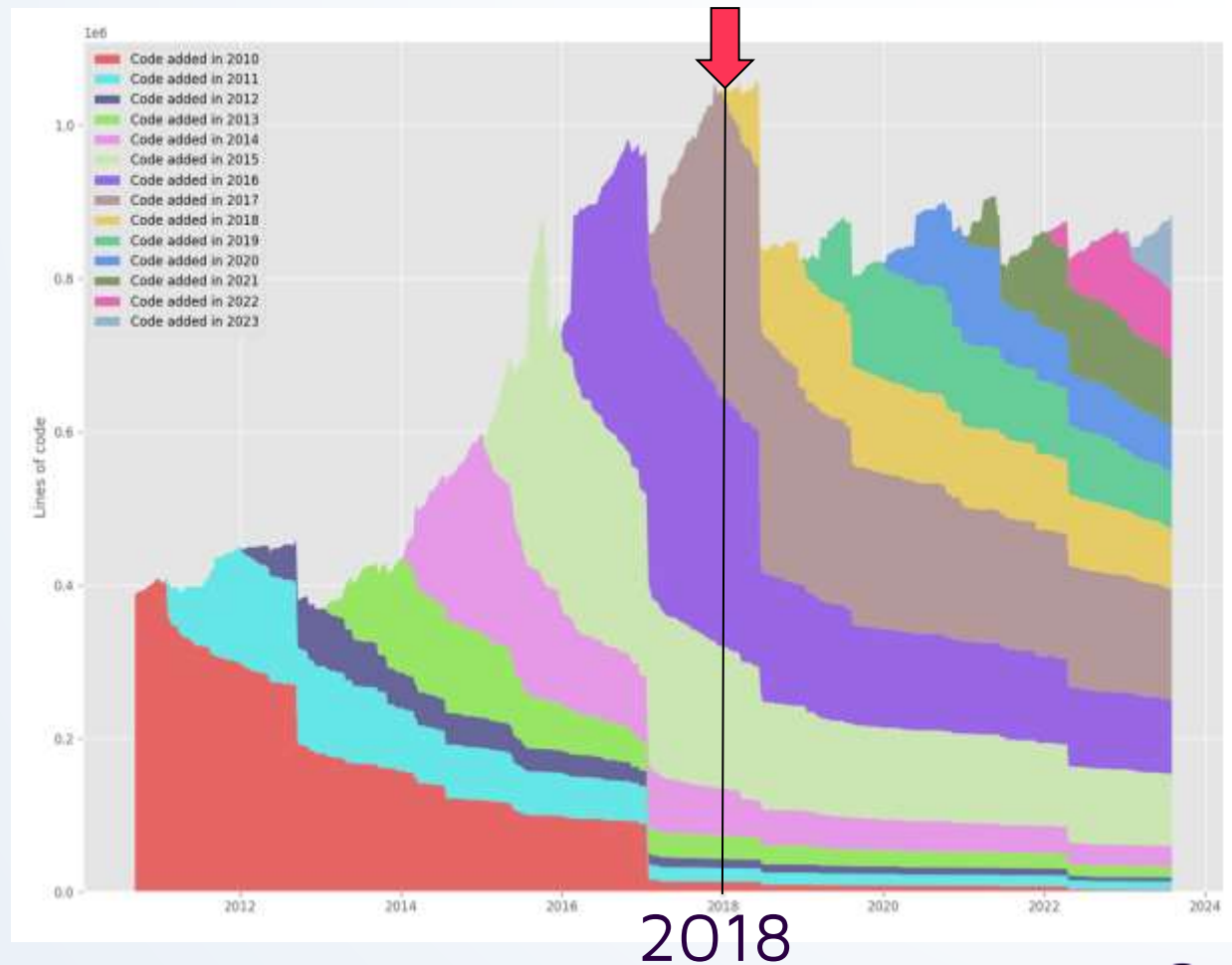
Here's SonarQube



Big delete
(bye-bye ruby code)

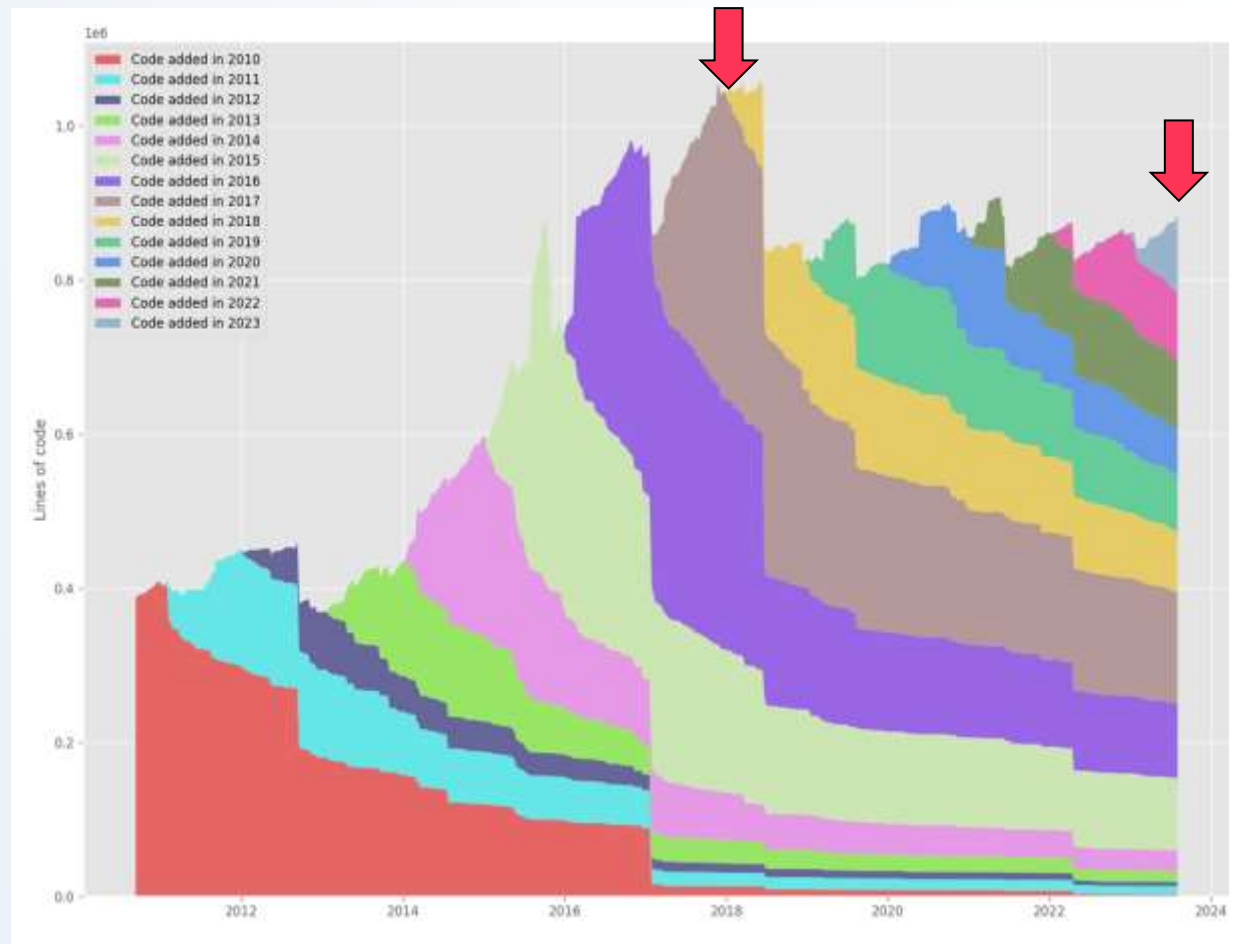
Here's SonarQube

At the
beginning of
2018 there
were 1 million
LOC



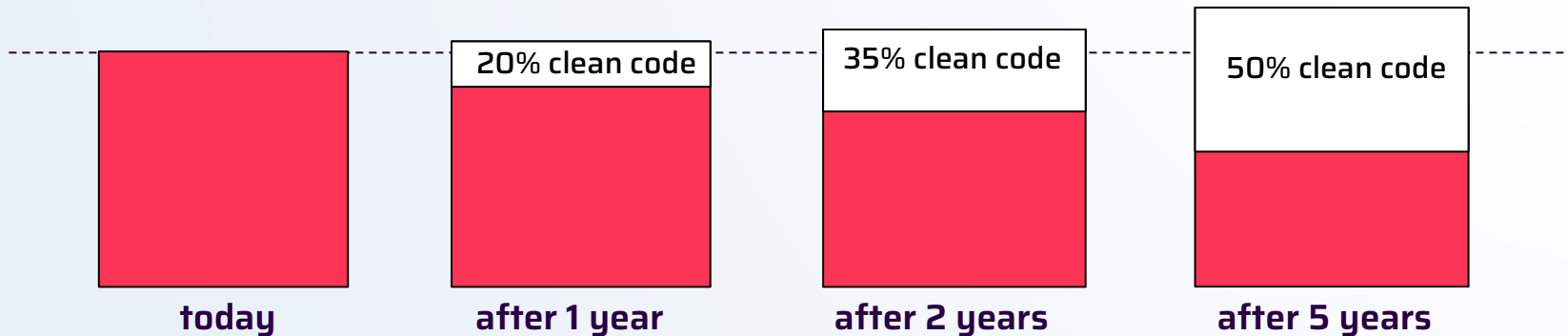
Here's SonarQube

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

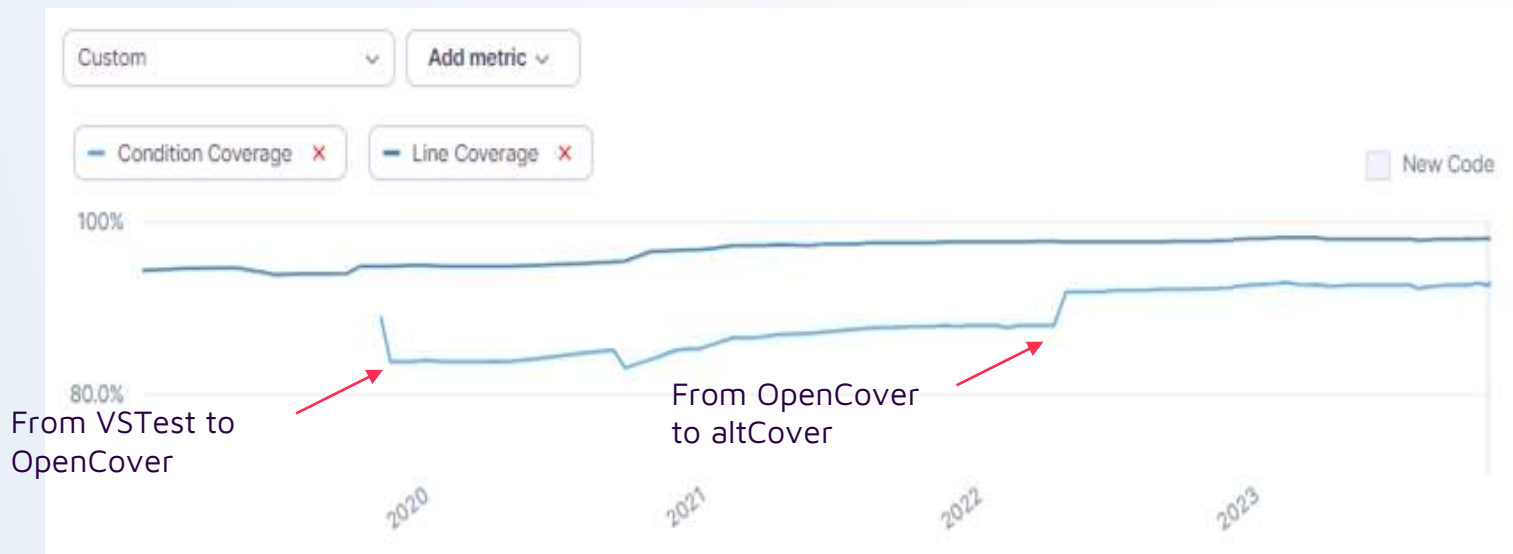
We don't merge PRs with red QG

Red QG = broken build (slack notification)

My experience at Sonar

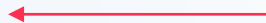
In three years, for  [sonar-dotnet](https://sonar-dotnet.com), we increased branch (conditional) coverage from 82% to 93% by using a Quality Gate at 95%.

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

Number of issues



Key takeaways

Remember this!



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

sonarlint



Free

for open source

sonarqube 
Community Edition

sonarcloud 

Import 3rd party Roslyn issues by default

Feedback form & slides:

 AndreiEpure.ro

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



T-SQL



COBOL



and
more...

