Доверяй, но проверяй

5 АНАЛИЗАТОРОВ КОДА НА КАЖДЫЙ ДЕНЬ

Алексей Чиркин

Senior Software Engineer (EPAM)

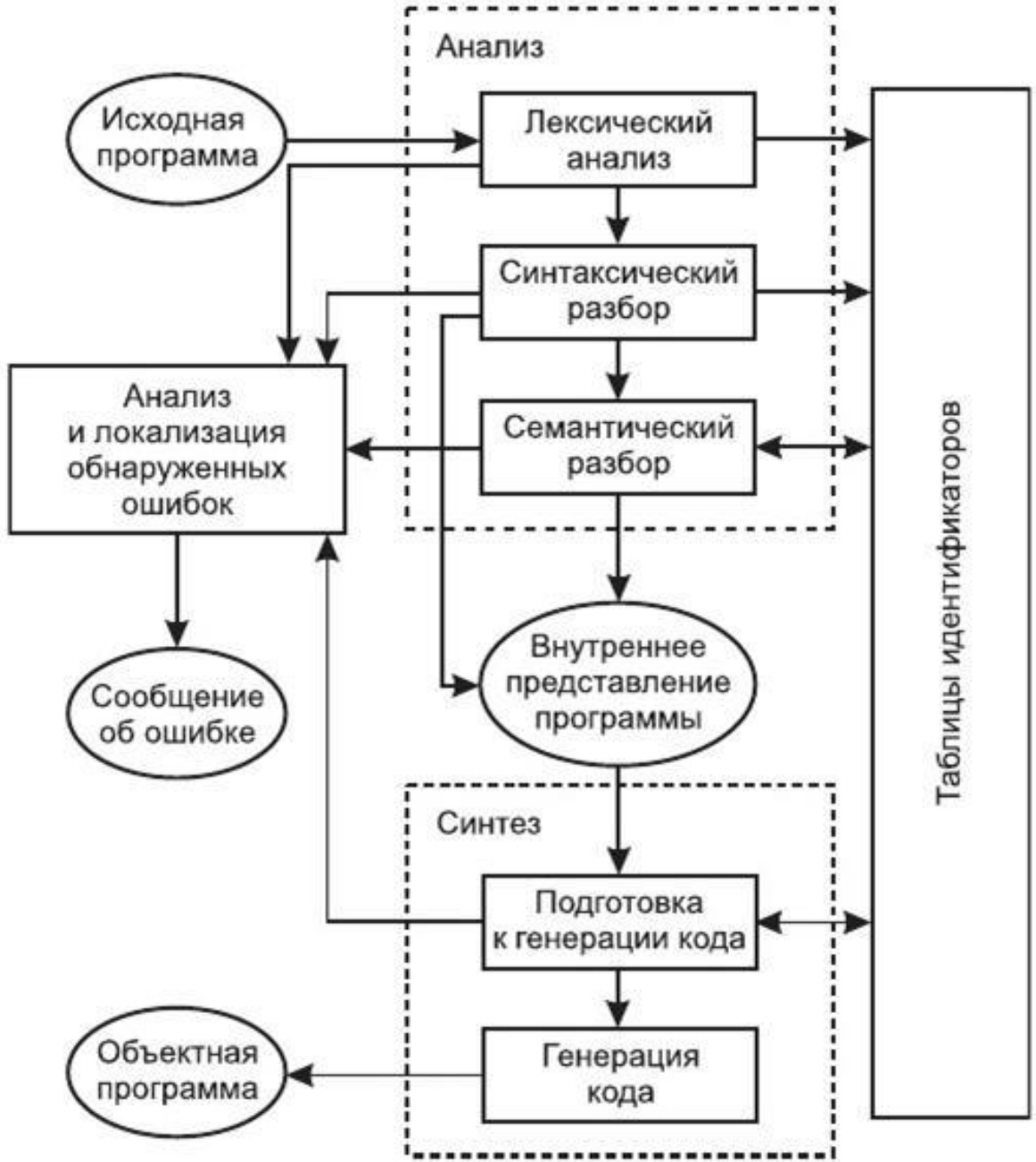
BIA Technologies Яндекс Деньги epam

Agenda

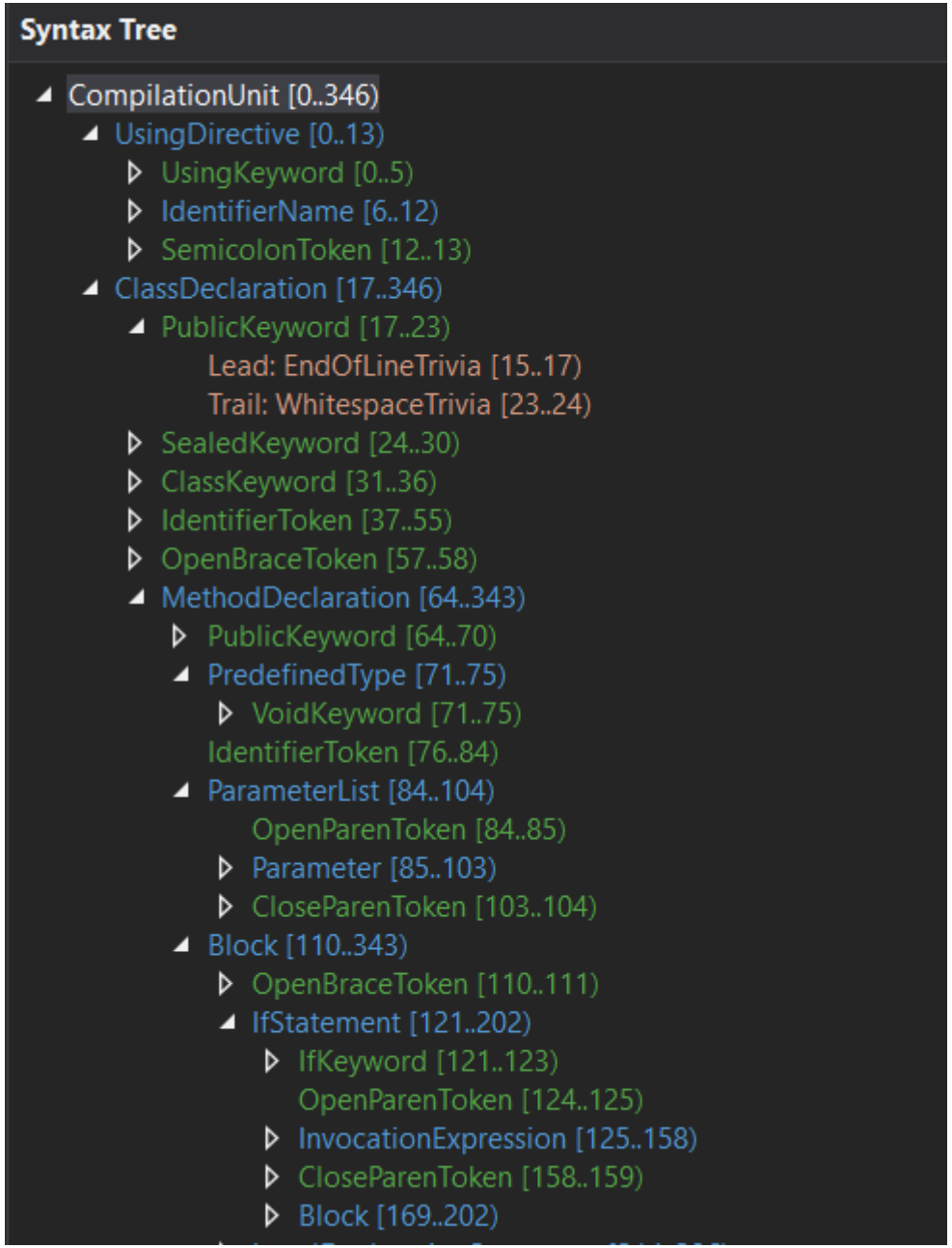
- О деревьях и прочем
- Не заблудимся в двух соснах
- Зачем нам всё это?
- Немного терминологии перед тем, как мы...
- ... отправимся в дебри
- Не Roslyn-ом единым...
- И ещё 4 анализатора кода на каждый день
- Благими намерениями устлана дорога... в прод!

О деревьях и прочем

```
using System;

public sealed class SoMysteriousForest
{
    public void CallEcho(string panicScream)
    {
        if (string.IsNullOrEmpty(panicScream))
        {
            return;
        }

        Console.WriteLine(panicScream);

        var echoAnswer = panicScream.Length <= 3 ? panicScream : panicScream.Substring(2);
        Console.WriteLine(echoAnswer);
    }
}
```


Анализ
Исходная программа
Лексический анализ
Синтаксический разбор
Семантический разбор
Анализ и локализация обнаруженных ошибок
Сообщение об ошибке
Внутреннее представление программы
Синтез
Подготовка к генерации кода
Генерация кода
Объектная программа
Таблицы идентификаторов

- **Лексический анализ** – разбор на лексемы → токены
- **Синтаксический анализ** (разбор) – токены → структура (дерево)
- **Семантический анализ** (разбор) – информация об объекте и его типе


Syntax Tree
CompilationUnit [0..346)
UsingDirective [0..13)
UsingKeyword [0..5)
IdentifierName [6..12)
SemicolonToken [12..13)
ClassDeclaration [17..346)
PublicKeyword [17..23)
Lead: EndOfLineTrivia [15..17)
Trail: WhitespaceTrivia [23..24)
SealedKeyword [24..30)
ClassKeyword [31..36)
IdentifierToken [37..55)
OpenBraceToken [57..58)
MethodDeclaration [64..343)
PublicKeyword [64..70)
PredefinedType [71..75)
VoidKeyword [71..75)
IdentifierToken [76..84)
ParameterList [84..104)
OpenParenToken [84..85)
Parameter [85..103)
CloseParenToken [103..104)
Block [110..343)
OpenBraceToken [110..111)
IfStatement [121..202)
IfKeyword [121..123)
OpenParenToken [124..125)
InvocationExpression [125..158)
CloseParenToken [158..159)
Block [169..202)

```
CompilationUnit()
.WithMembers(
    SingletonList<MemberDeclarationSyntax>(
        MethodDeclaration(
            IdentifierName("Task"), Identifier("RegisterCodeFixesAsync"))
.WithModifiers(
    TokenList(
        new []{
            Token(SyntaxKind.PublicKeyword),
            Token(SyntaxKind.SealedKeyword),
            Token(SyntaxKind.OverrideKeyword),
            Token(SyntaxKind.AsyncKeyword)}))
.WithParameterList(
    ParameterList(
        SingletonSeparatedList<ParameterSyntax>(
            Parameter(
                Identifier("context"))
.WithType(
    IdentifierName("CodeFixContext")))))

… over 100 rows of panic scream code…
```

https://github.com/KirillOsenkov/RoslynQuoter

- Для обработки деревьев существует множество алгоритмов
- Деревья являются более «понятными», чем бесконечное множество неструктурированных лексем
- Деревья удобны для дальнейших преобразований

Не заблудимся в двух соснах

FxCop (aka Run Code Analysis)

- Легаси-подход к расширению возможностей статического анализа
- Работает с уже скомпилированным кодом
- Представляет из себя набор правил в файлах ruleset
- Вкладка “Code Analysis” в параметрах проекта Visual Studio – это про него

FxCop (aka Roslyn) Analyzers

- Новый подход, базирующийся на API Roslyn;
- Работает во время написания кода
- Распространяется как nuget-пакет или как VSIX
- Warnings про то, что Run Code Analysis был признан устаревшим в пользу FxCop – это про него

https://docs.microsoft.com/en-gb/dotnet/csharp/roslyn-sdk/tutorials/how-to-write-csharp-analyzer-code-fix

Зачем нам всё это?

Maintainability

- Очистить код от визуального мусора (повторяющиеся последовательности литералов, невалидные имена переменных и пр.)
- Отслеживать объём комментариев в коде

Security

- Ограничивать вызовы нежелательных методов сторонних библиотек, даже если они не являются Obsolete
- Предупреждать о вызовах методов с реализацией «по умолчанию» (содержащих выброс NotImplementedException)
- Запрещать «хардкод» полей, содержащих конфиденциальную информацию

Performance

- Предлагать заменять «тяжёлые» строковые операции (String.Substring()) на использование Span<string>.Slice()
- Предлагать заменять полное чтение из потока с непредсказуемым объемом данных на использование yield

Немного терминологии перед тем, как мы...

Syntax nodes – объявления, операторы, выражения и т.д. (SyntaxNode)

Syntax tokens (лексемы) – идентификаторы, ключевые слова, спецсимволы и т.д. (SyntaxToken)

Syntax trivia – дополнительная синтаксическая информация – пробелы, символы перевода строки, комментарии, директивы препроцессора и т.д. (SyntaxTrivia)

https://habr.com/ru/company/pvs-studio/blog/301204/

Term	Description
Analyzer	Represents a general rule (code style) that should be followed.
Diagnostic	Represents a specific issue reported by the analyzer.
Code Fix	Represents an operation that will fix reported issue.
Refactoring	Represents a single operation that is provided on demand for a given span of text.

https://github.com/JosefPihrt/Roslynator/blob/master/docs/AnalyzersVsRefactorings.md

… отправимся в дебри

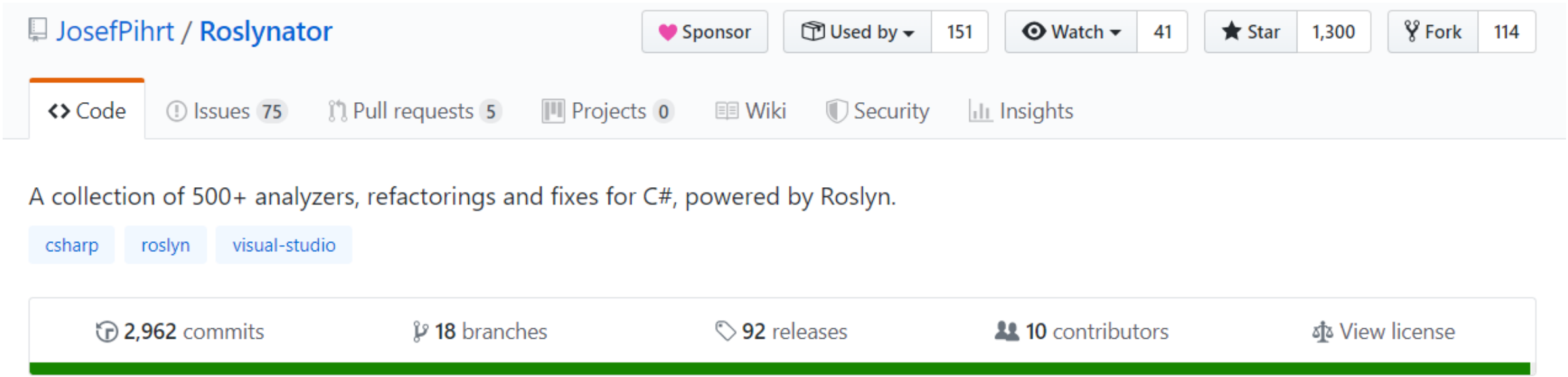
DEMO

Не Roslyn-ом единым…

- Roslyn – **чрезвычайно мощный** инструмент, но за мощность приходится платить **сложностью**.
- Анализаторы, написанные на «голом» Roslyn API, **нужно** серьёзно **«докручивать»** с помощью множества юнит-тестов, покрывающих максимальное число кейсов.
- Code fixes, базирующиеся на Roslyn API, порой бывают крайне **многословны**.
- **Необходимо** постоянно **мониторить прозводительность** анализаторов – в противном случае, user experience использования IDE может сильно пострадать.

Roslynator

A collection of 500+ analyzers, refactorings and fixes for C#, powered by Roslyn.



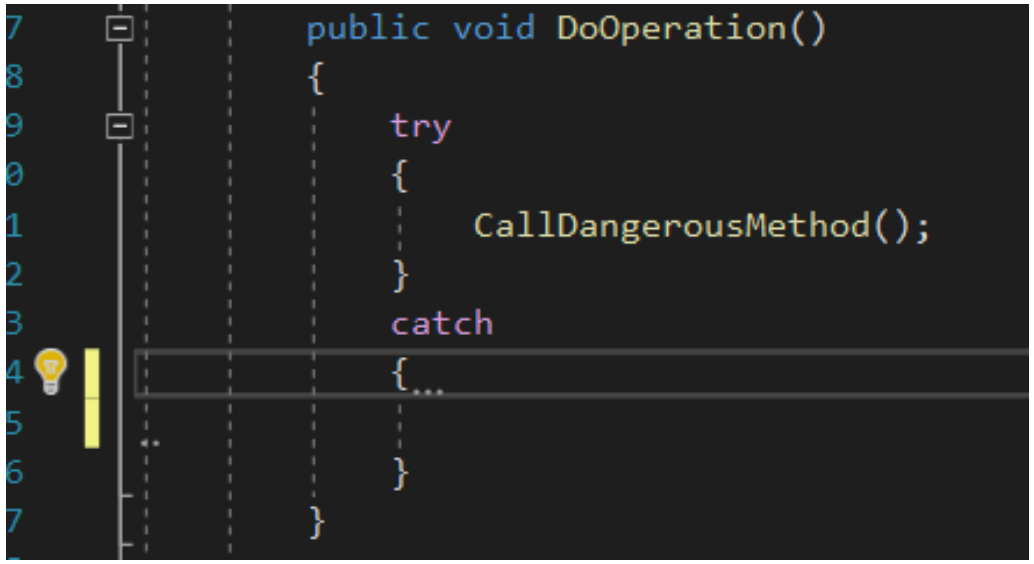
https://github.com/JosefPihrt/Roslynator

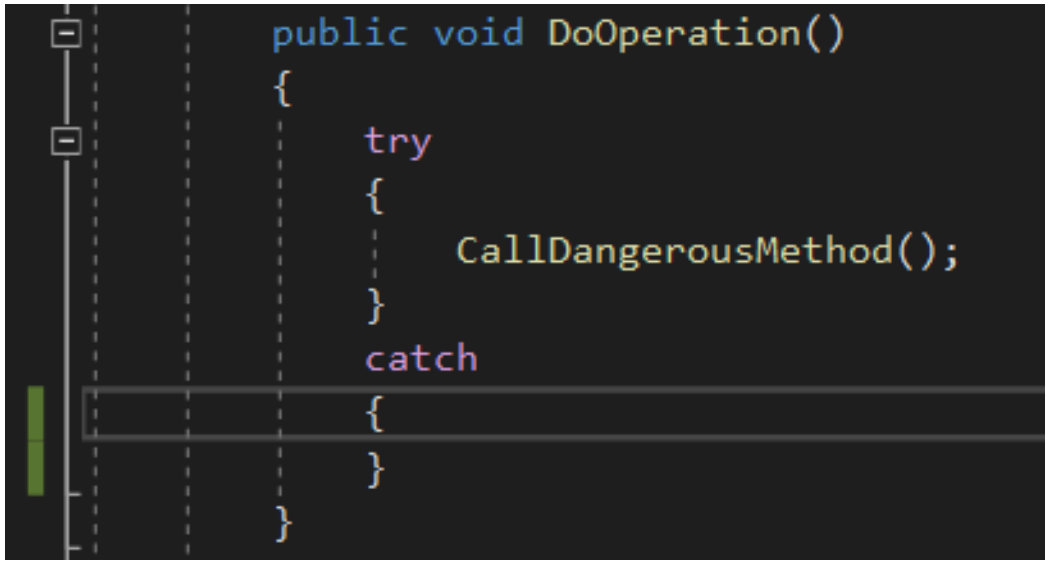
И ещё 4 анализатора кода на каждый день

- Актуальны для каждодневного использования
- Бесплатны
- С открытым исходным кодом
- Отсутствуют в Roslyn Analyzers
- Достаточно просты для знакомства с дебрями Roslyn

Whitespace analyzer

Удаляет лишние пробелы и пустые строки

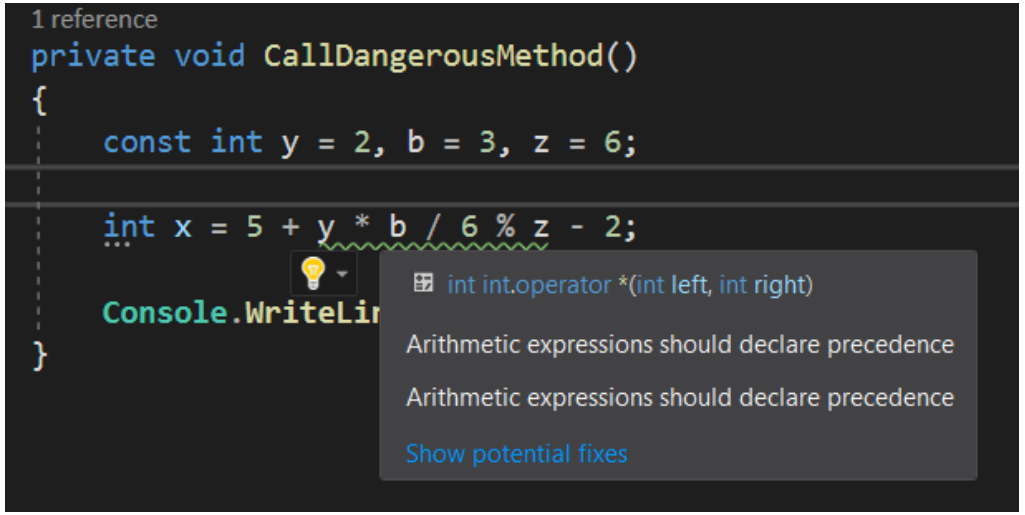


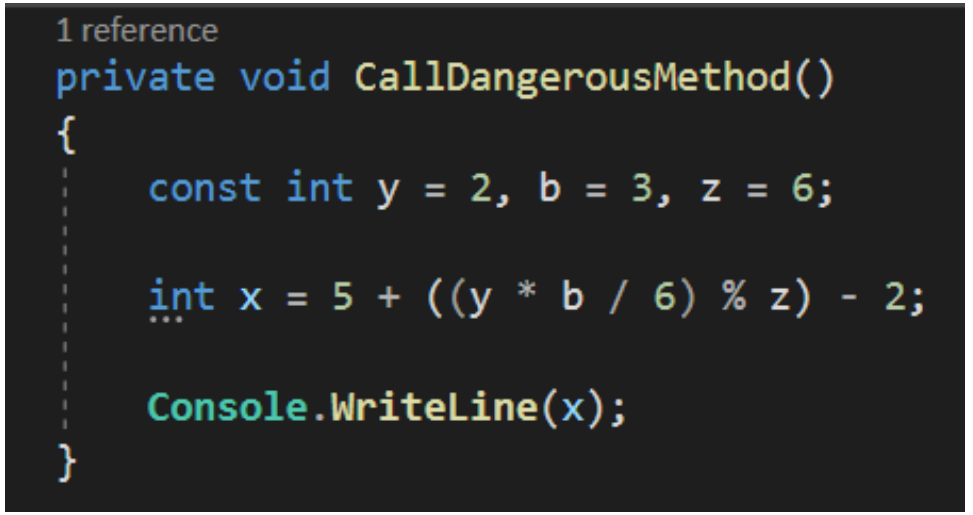


https://github.com/JosefPihrt/Roslynator/blob/master/src/Analyzers/Analysis/WhitespaceAnalyzer.cs

Arithmetics expression analyzer

Разбивает арифметическое выражение на группы с помощью скобок

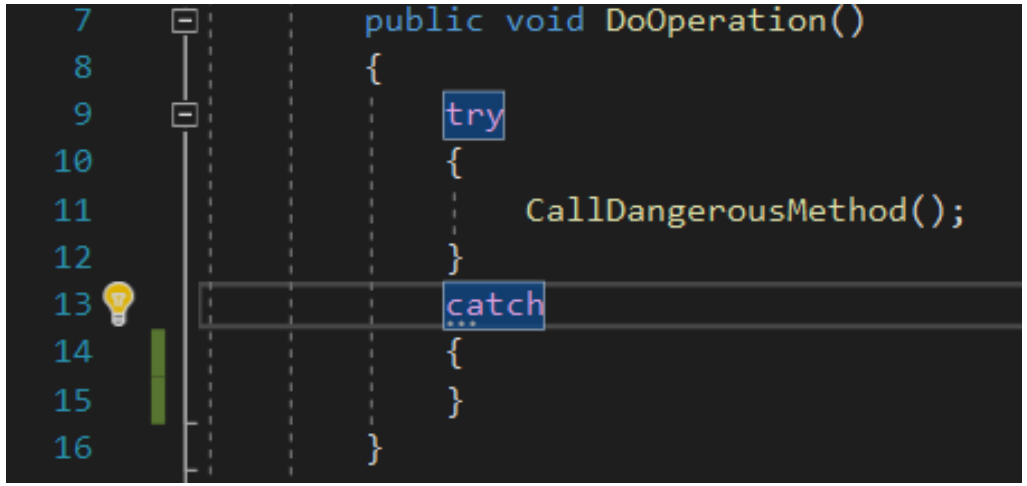


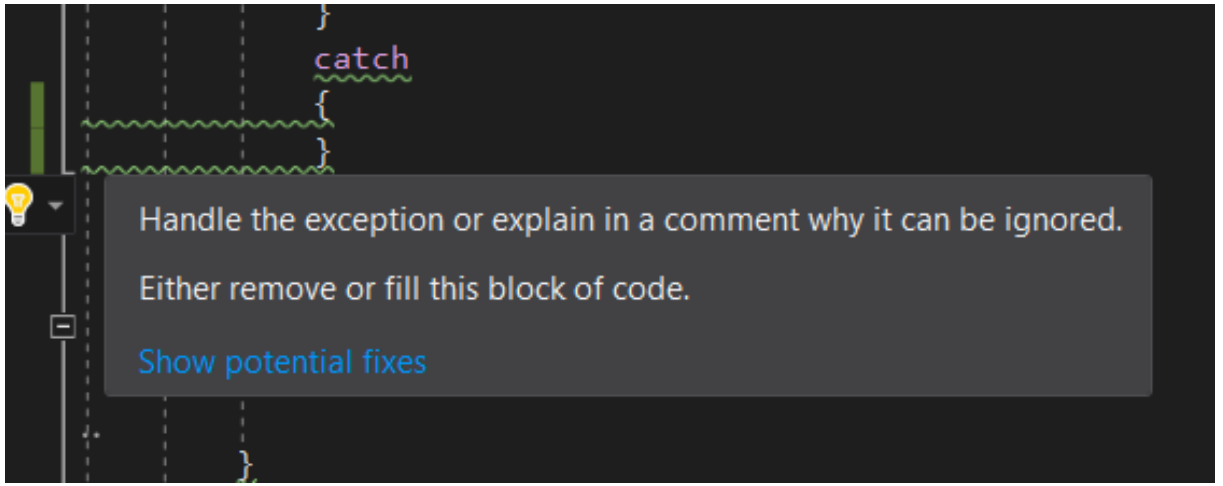


https://github.com/DotNetAnalyzers/StyleCopAnalyzers/blob/master/StyleCop.Analyzers/StyleCop.Analyzers/MaintainabilityRules/SA1407ArithmeticExpressionsMustDeclarePrecedence.cs

Catch blocks analyzer

Ищет пустые catch-блоки, «проглатывающие» исключения (diag only)



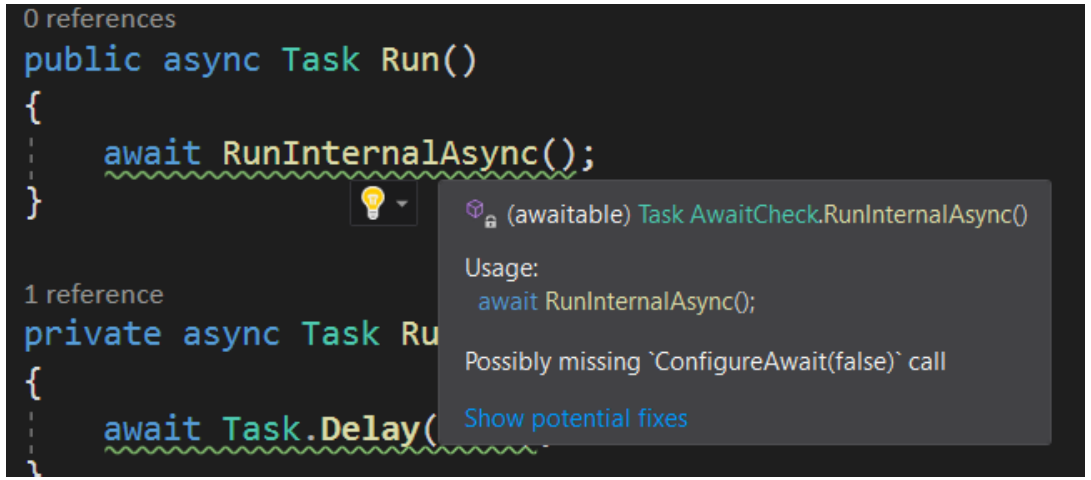


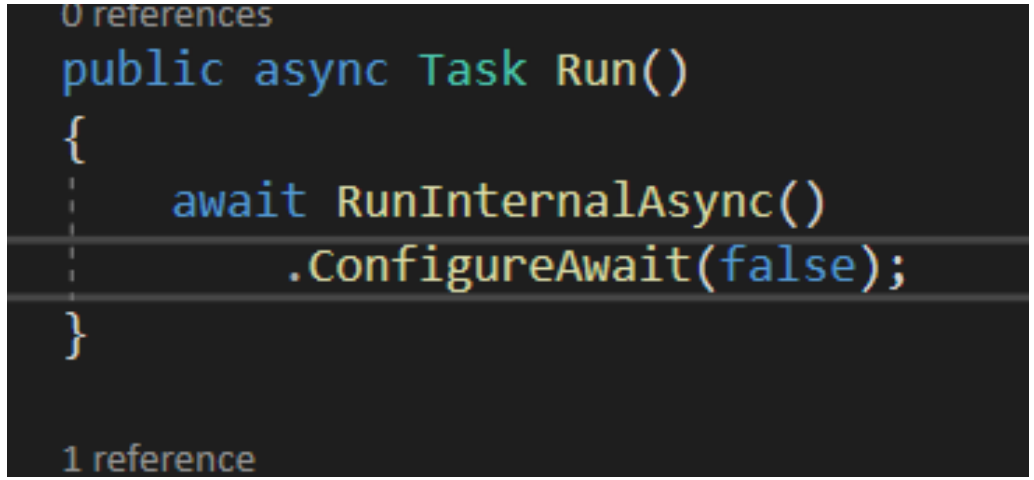
https://github.com/Wintellect/Wintellect.Analyzers/blob/master/Source/Wintellect.Analyzers/Wintellect.Analyzers/Design/CatchBlocksShouldRethrowAnalyzer.cs

https://rules.sonarsource.com/csharp/RSPEC-2486

ConfigureAwait checker

Ищет и помогает исправить (опционально) асинхронные вызовы методов





https://github.com/cincuranet/ConfigureAwaitChecker

https://github.com/JosefPihrt/Roslynator/blob/2f7c85d35aa65d4e3243c13a97715f0a5a17d6c7/docs/analyzers/RCS1090.md

Благими намерениями устлана дорога... в прод!

- Статические анализаторы позволяют расширить проверку кода так, как это нужно Вам
- Если Вам не хватает возможностей, предоставляемых анализаторами компании Microsoft – есть альтернативы
- Если альтернатив найти не удалось или они Вас не устраивают – напишите свой анализатор!
- С помощью анализаторов можно не только исправлять небрежный код и находить потенциальные баги. С помощью Roslyn API можно писать валидации для собственной бизнес-логики, связанные с особенностями именно Ваших бизнес-процессов!
- Написание анализаторов кода позволяет лучше разобраться в том, как работает сам код

Полезные ссылки

Репозиторий с примером из демо - https://github.com/avchirkin/EverydayAnalyzers

Roslyn Analysers - https://github.com/dotnet/roslyn-analyzers

Roslynator - https://github.com/JosefPihrt/Roslynator

RoslynQuoter - https://github.com/KirillOsenkov/RoslynQuoter

StyleCop - https://github.com/DotNetAnalyzers/StyleCopAnalyzers

Книга дракона - http://www.williamspublishing.com/Books/978-5-8459-1349-4.html

Группа VK SamDotNet – https://vk.com/samdotnet

Telegram-канал SamDotNet – https://t.me/samdotnet

Telegram-чат SamDotNet – https://t.me/samdotnetchat

Спасибо за внимание!