Не все статические анализаторы одинаково полезны

Владимир Кошелев

ИСП РАН

План доклада

1. Поиск ошибочных шаблонов
   a) Компиляторная инфраструктура Roslyn
   b) Обзор инструментов для поиска ошибок: Roslyn, ReSharper, PVS-Studio
   c) Сравнение Roslyn, ReSharper, PVS-Studio

План доклада

1. Поиск ошибочных шаблонов
   a) Компиляторная инфраструктура Roslyn
   b) Обзор инструментов для поиска ошибок: Roslyn, ReSharper, PVS-Studio
   c) Сравнение Roslyn, ReSharper, PVS-Studio
2. Поиск сложных ошибок
   a) Почему unit-тестирования недостаточно
   b) Динамическое символьное выполнение IntelliTest (Pex)
   c) Статическое символьное выполнение

План доклада

1. Поиск ошибочных шаблонов
   a) Компиляторная инфраструктура Roslyn
   b) Обзор инструментов для поиска ошибок: Roslyn, ReSharper, PVS-Studio
   c) Сравнение Roslyn, ReSharper, PVS-Studio
2. Поиск сложных ошибок
   a) Почему unit-тестирования недостаточно
   b) Динамическое символьное выполнение IntelliTest (Pex)
   c) Статическое символьное выполнение
3. Интеграция в процесс разработки

Как используются анализаторы

- Поиск багов до запуска программы

Как используются анализаторы

- Поиск багов до запуска программы
- Рефакторинг

Как используются анализаторы

- Поиск багов до запуска программы
- Рефакторинг
- Coding guidelines & Best Practices

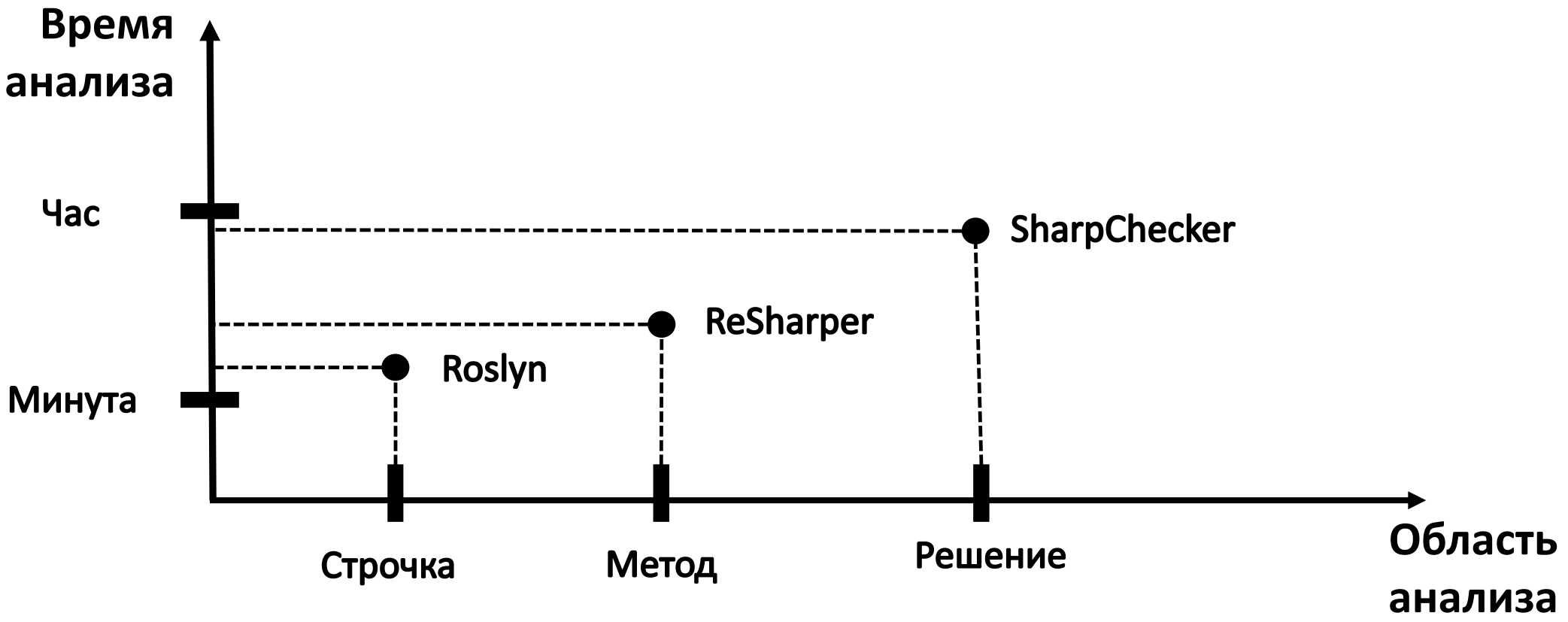
Как используются анализаторы

- Поиск багов до запуска программы
- Рефакторинг
- Coding guidelines & Best Practices
- Code-aware libraries

Как используются анализаторы

- Поиск багов до запуска программы
- Рефакторинг
- Coding guidelines & Best Practices
- Code-aware libraries
- Обучение новым фичам языка

Разные области анализа для поиска ошибок



.NET Compiler Platform “Roslyn”

- Открытая платформа для разработки статических анализаторов

.NET Compiler Platform “Roslyn”

- Открытая платформа для разработки статических анализаторов

- Удобное API:
  - Workspace, Solution, Project
  - Syntax Tree & Semantic Model
  - IOperation
  - Code Fixes

.NET Compiler Platform “Roslyn”

- Открытая платформа для разработки статических анализаторов
- Удобное API:
  - Workspace, Solution, Project
  - Syntax Tree & Semantic Model
  - IOperation
  - Code Fixes
- Автоматическая интеграция в процесс разработки

Полезные пакеты анализаторов

SonarLint	SonarSource-VisualStudio/**sonaranalyzer-csharp**
Roslyn-analyzers	dotnet/**roslyn-analyzers**
StyleCop	DotNextAnalyzers/**StyleCopAnalyzers**
Code Cracker for C#	code-cracker/**code-cracker**
Heap Allocation Analyzer	mjsabby/**RoslynClrHeapAllocationAnalyzer**

Анализаторы Roslyn для поиска ошибок

- Категории Reliability и Security

Анализаторы Roslyn для поиска ошибок

- Категории Reliability и Security

- К сожалению, анализаторы группы Reliability действительно хорошо представлены только в SonarLint

Анализаторы Roslyn для поиска ошибок

- Категории Reliability и Security
- К сожалению, анализаторы группы Reliability действительно хорошо представлены только в SonarLint
- Итого: около 80 типов предупреждений из Reliablity.

ReSharper для поиска ошибок

- Две категории предупреждений:
  - Potential Code Quality Issue
  - Redundancies in Code

ReSharper для поиска ошибок

- Две категории предупреждений:
  - Potential Code Quality Issue
  - Redundancies in Code

- Для языка C# суммарно в этих категориях около 170 типов предупреждений

ReSharper для поиска ошибок

- Две категории предупреждений:
  - Potential Code Quality Issue
  - Redundancies in Code

- Для языка C# суммарно в этих категориях около 170 типов предупреждений

- Не все типы предупреждений указывают на потенциальные ошибки

PVS-Studio

- Инструмент специализируется именно на поиске ошибок

PVS-Studio

- Инструмент специализируется именно на поиске ошибок
- Хорошая документация

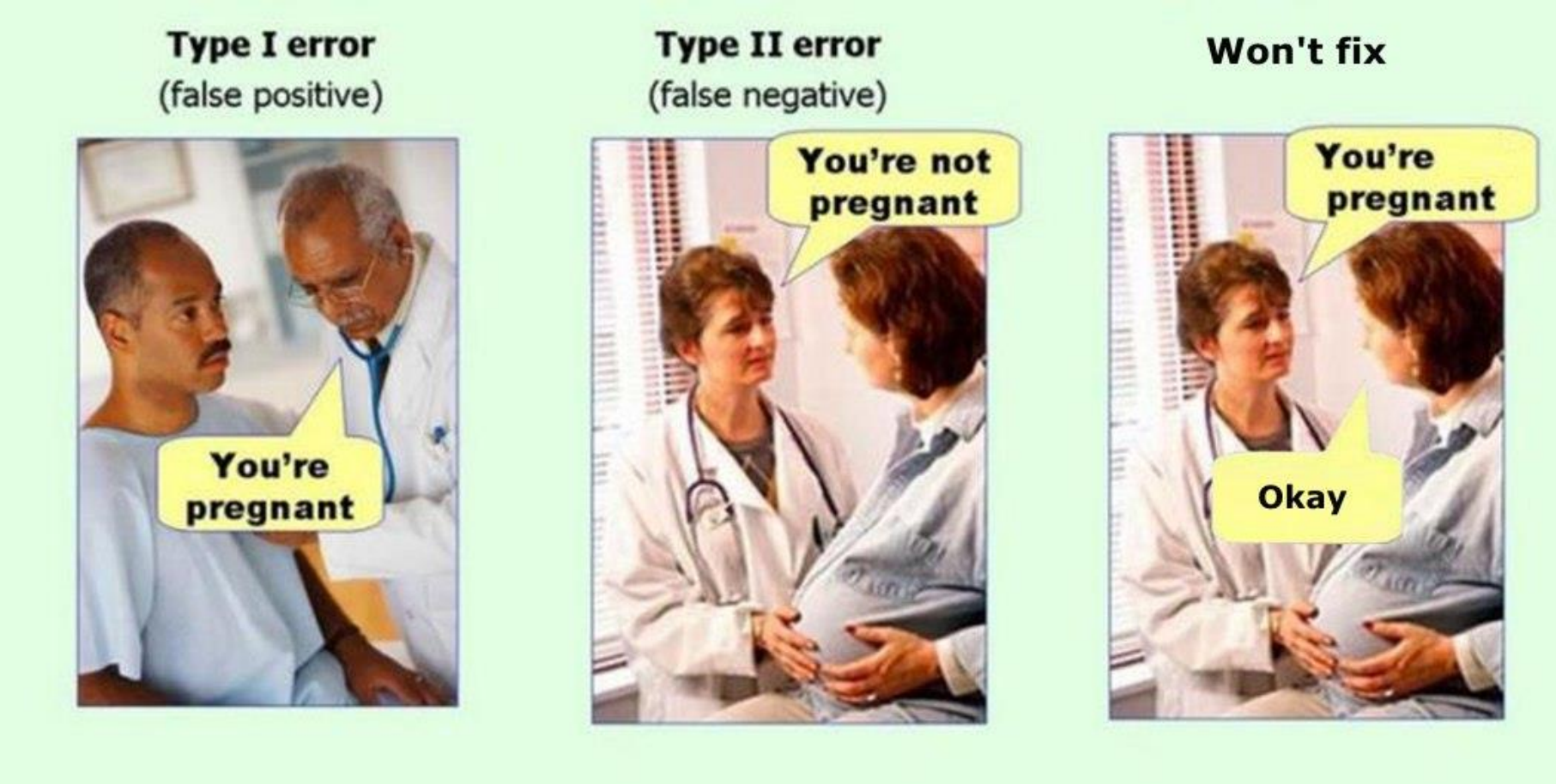
PVS-Studio

- Инструмент специализируется именно на поиске ошибок
- Хорошая документация
- На данный момент поддерживается 93 предупреждения

PVS-Studio

- Инструмент специализируется именно на поиске ошибок
- Хорошая документация
- На данный момент поддерживается 93 предупреждения
- Особый акцент на поиске опечаток/ошибок copy-paste и т.д.

Ложные срабатывания и пропуски ошибок



Как сравнивать разные инструменты

- Цель: понять какие ошибки находят инструменты и как они пересекаются

Как сравнивать разные инструменты

- Цель: понять какие ошибки находят инструменты и как они пересекаются

- **Как**: берем много разных OpenSource проектов, запускаем на них все анализаторы

Как сравнивать разные инструменты

- Цель: понять какие ошибки находят инструменты и как они пересекаются

- **Как**: берем много разных OpenSource проектов, запускаем на них все анализаторы

- Ищем совпадения предупреждений разных инструментов по строкам. Среди пар совпавших предупреждений оставляем те, которые относятся к схожим типам.

Какие проекты анализировались?

- blogengine
- BobBuilder
- CodeContracts
- codeplexdlr
- cosmos
- csharp-driver
- Csharputils
- csparser
- dynamicrest
- Jil
- Libgit2sharp
- lucenenet
- mailsystem
- mathos
- nant
- netmq
- openbve
- Polly
- sake
- ShareX
- susanoo
- wcell

Всего 22 проекта, 3 000 000 строк кода

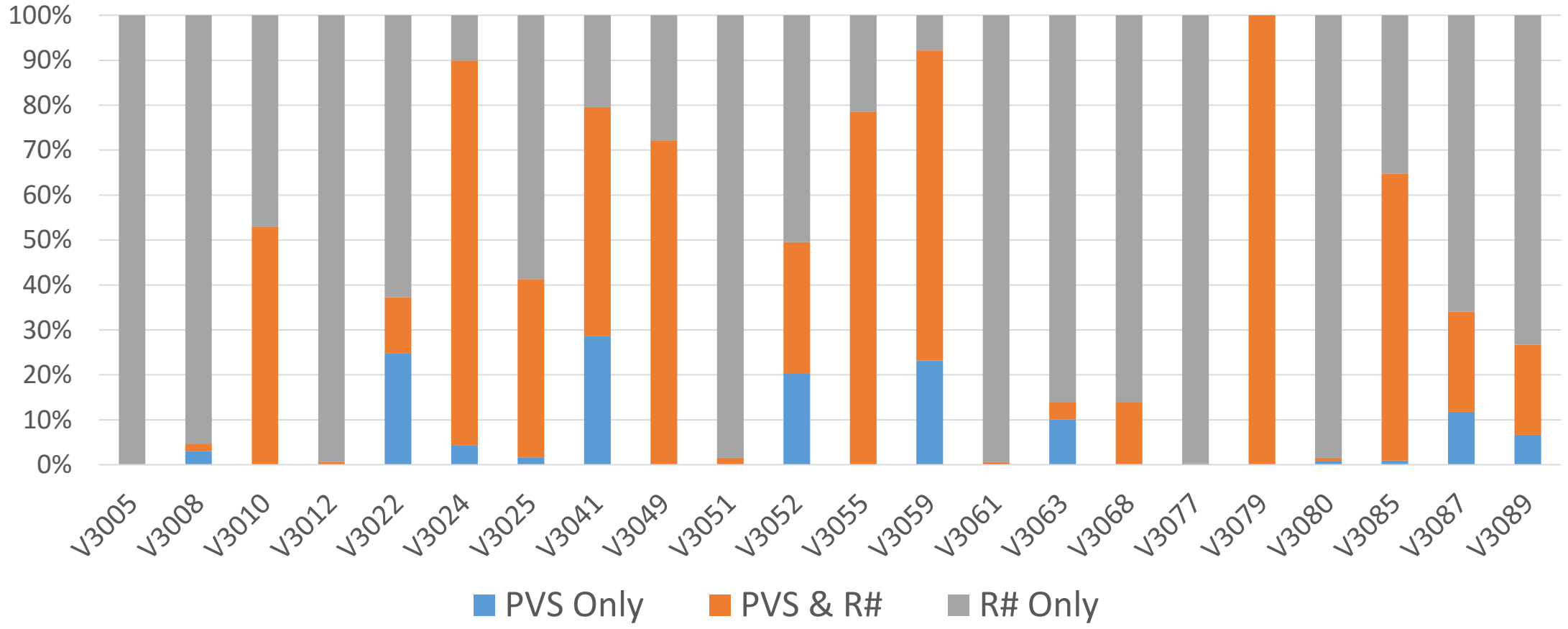
Результаты: сколько всего срабатываний

Название инструмента	Сколько предупреждений	Типов покрыто	Типов всего
SonarLint	11500	48	80
PVS-Studio	3150 + 2600*	67	93
Resharper	40000 + 150000**	126	170

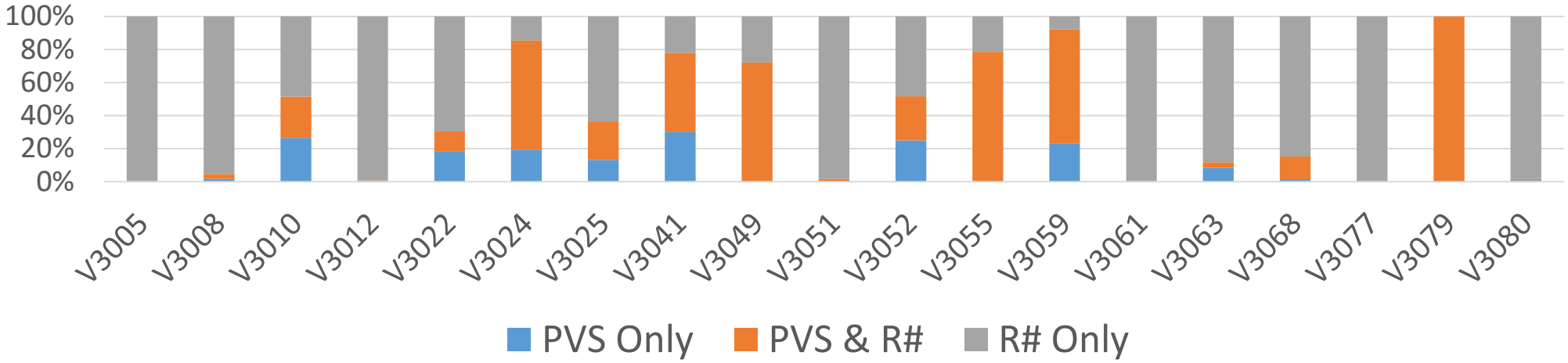
*У PVS-Studio 1100 предупреждений приходится на **floating point comparison**, и ещё 1500 на **static readonly uninitialized DependencyProperty**

** 150 000 предупреждений ReSharper из типов **RedundantNameQualifier**, **RedundantUsingDirective** и **ObjectCreatingAsStatement** проигнорированы

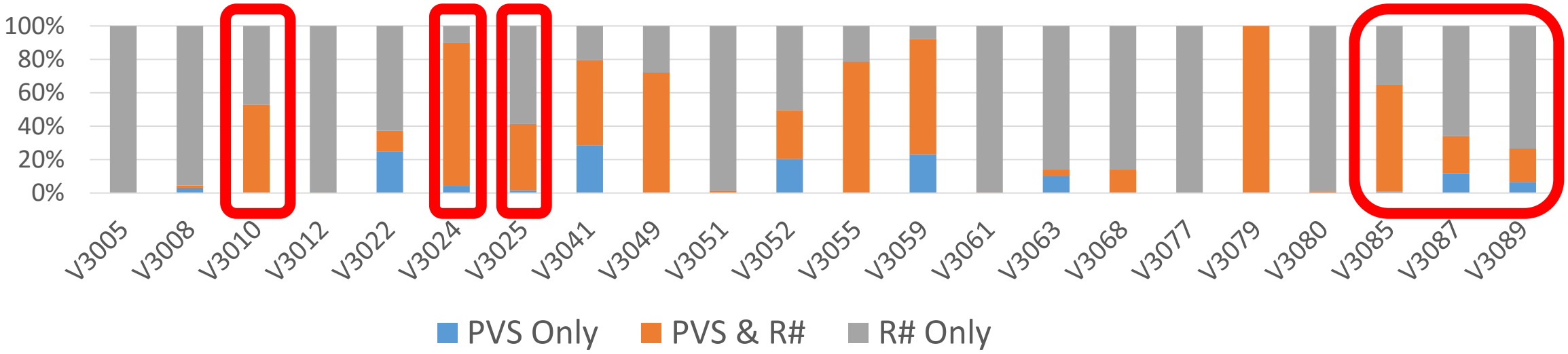
PVS 6.04 / ReSharper



PVS 6.03 / ReSharper



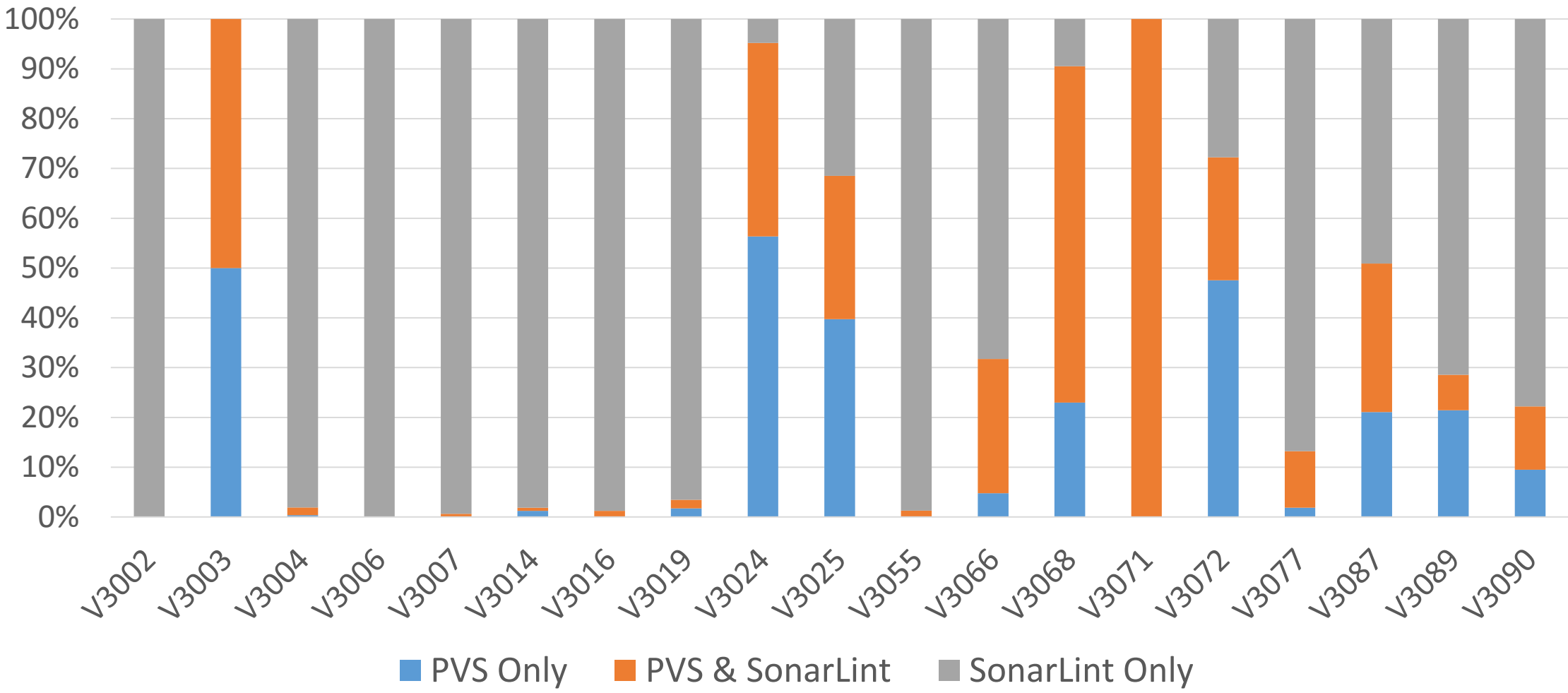
PVS 6.04 / ReSharper



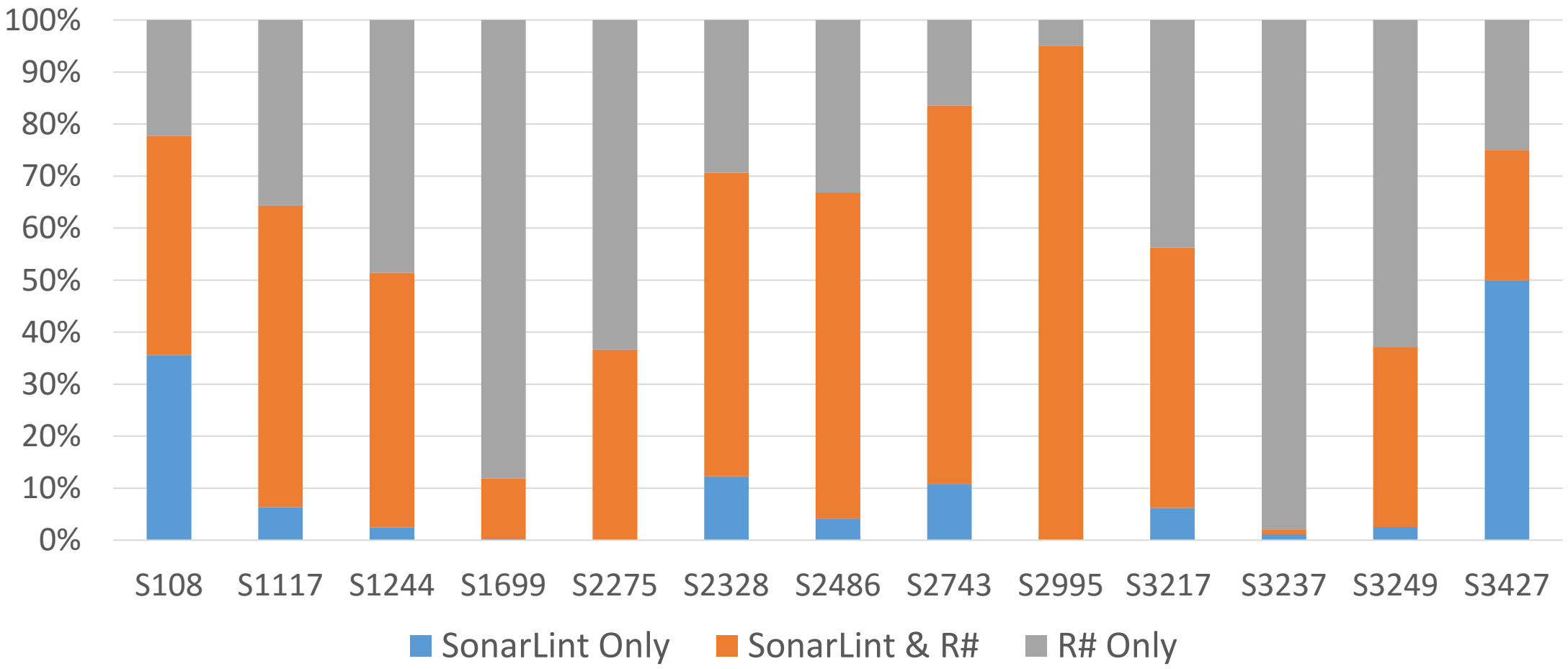
Пример: разные категории, но одна ошибка

```
if (origNode == null) {
     ...
     ReSharper: Expresion origNode is always null.
     return (origNode == null) ? 1 : 2;
}
---------------------
if (origNode == null) {
     ...
     PVS-Studio: Expression (origNode == null) ? 1 : 2 always has the same value.
     return (origNode == null) ? 1 : 2;
}
```

PVS 6.04 /SonarLint



SonarLint / ReSharper



Выводы

На вкус и цвет все фломастеры разные

Выводы

На вкус и цвет все фломастеры разные

Resharper и SonarLint находят больше

PVS находит меньше, но куда чаще это баги

Можно использовать всех вместе

Часть 2. Поиск сложных ошибок

- Что такое сложные ошибки:
  - Возможные NullReferenceException
  - Утечка ресурсов
  - Использование disposed объектов
  - Возможное деление на 0
  - Уязвимости приводящие к Injection-атакам

Часть 2. Поиск сложных ошибок

- Что такое сложные ошибки:
  - Возможные NullReferenceException
  - Утечка ресурсов
  - Использование disposed объектов
  - Возможное деление на 0
  - Уязвимости приводящие к Injection-атакам

- Ключевая особенность:

  Есть причина возникновения ошибки, и есть место проявления.

Тестирование и статический анализ ортогональны

- Unit testing рассматривает лишь определенный набор ситуаций
- Статический анализ смотрит на ситуации в целом
- Даже 100% Coverage не означает отсутствие ошибок

Почему block coverage недостаточно

```
public static bool ProcessDataWrapper(Data data, Processor processor) {
     if (processor == null && data == null)
          return false;

     return processor.ProcessData(data);
}
```

Почему block coverage недостаточно

```
public static bool ProcessDataWrapper(Data data, Processor processor) {
     if (processor == null && data == null)
          return false;

     return processor.ProcessData(data);
}
```

Почему block coverage недостаточно

```
public static bool ProcessDataWrapper(Data data, Processor processor) {
     if (processor == null && data == null)
          return false;

     return processor.ProcessData(data);
}

Assert.False(ProcessDataWrapper(null, null));
```

Почему block coverage недостаточно

```
public static bool ProcessDataWrapper(Data data, Processor processor) {
     if (processor == null && data == null)
          return false;

     return processor.ProcessData(data);
}

Assert.False(ProcessDataWrapper(null, null));
Assert.True(ProcessDataWrapper(new Data(), new Processor()));
```

WhiteBox testing [dynamic symbolic execution]

- Идея: вместо того, чтобы писать тесты самому, давайте генерировать их автоматически

WhiteBox testing [dynamic symbolic execution]

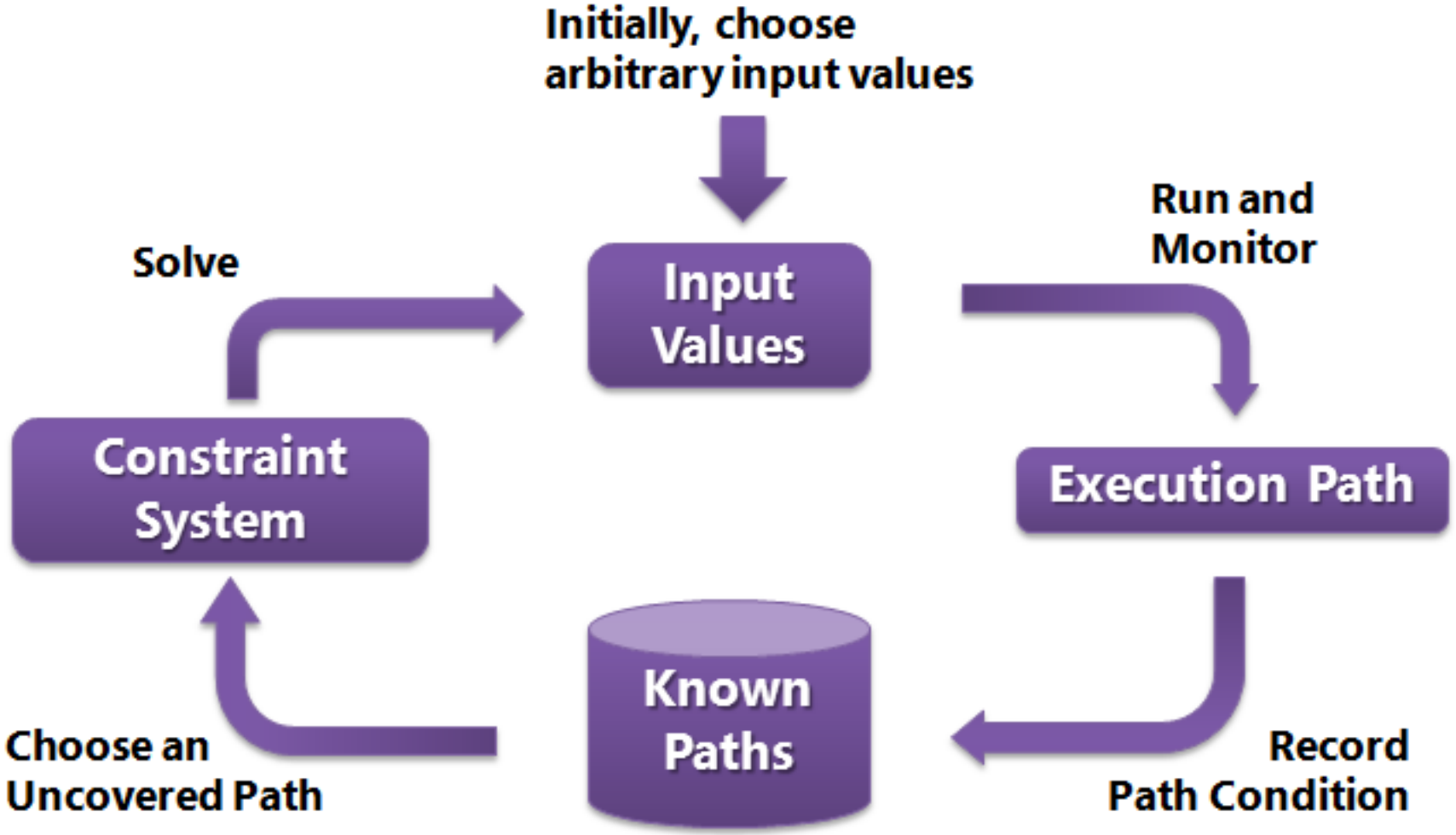
- Идея: вместо того, чтобы писать тесты самому, давайте генерировать их автоматически

- Решение от Mircosoft : IntelliTest (Visual Studio 2015 Enterprise)

WhiteBox testing [dynamic symbolic execution]

- Идея: вместо того, чтобы писать тесты самому, давайте генерировать их автоматически
- Решение от Mircosoft : IntelliTest (Visual Studio 2015 Enterprise)
- Публикация и proof of concept – 2008 год. [http://pex4fun.com/]
- Продакшн решение – 2015

Две виртуальные машины: одна обычная, вторая для символьного выполнения



Пример: ищем коллизии в CRC64

```
public static void Puzzle(byte[] bytes)
{
     if (Crc64Iso3309.Compute(bytes) == 0x1234567890abcdefL)
         throw new NullReferenceException();
}

protected static UInt64 Compute(IList<byte> buffer) {
      var crc = 0;
      for (var i = 0; i < buffer.Count; i++)
           unchecked {
                crc = (crc >> 8) ^ table[(buffer[i] ^ crc) & 0xff];
           }
      return crc;
}
```

Пример: ищем коллизии в CRC64

```
public static void Puzzle(byte[] bytes)
{
     if (Crc64Iso3309.Compute(bytes) == 0x1234567890abcdefL)
         throw new NullReferenceException();
}
```

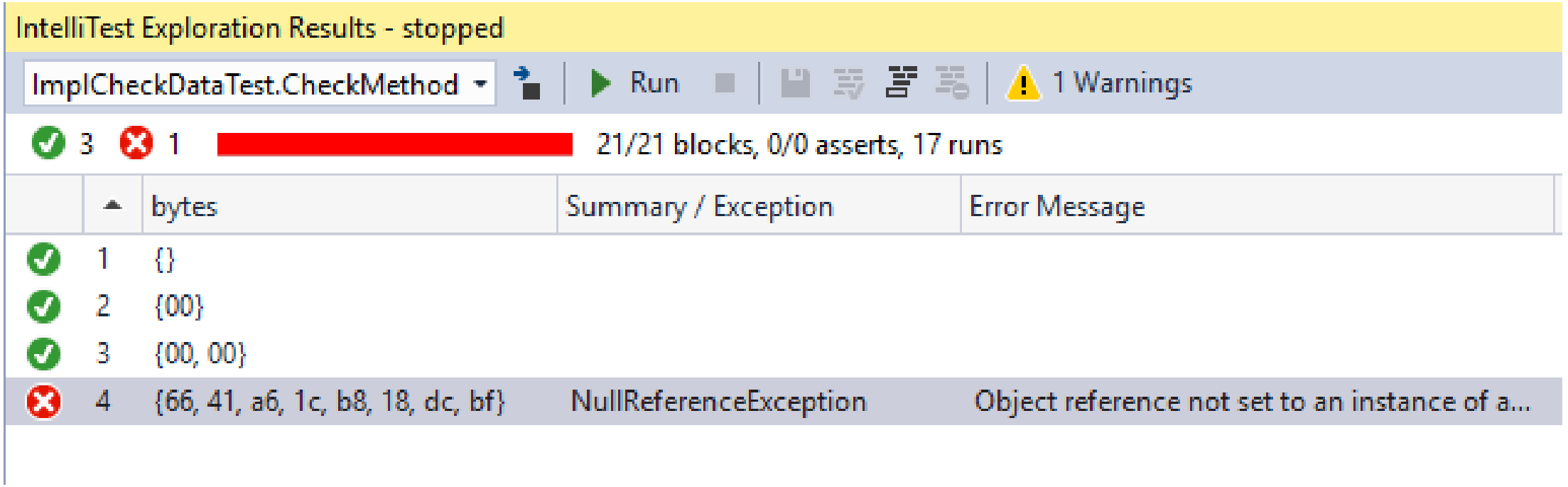
Пример: ищем коллизии в CRC64

```
[PexMethod]
public void CheckMethod(byte []bytes)
{
     PexAssume.IsNotNull(bytes);
     ImplCheckData.Crc32.Puzzle(bytes);
}
```

Пример: ищем коллизии в CRC64

```
[PexMethod(MaxConstraintSolverTime = 1200, Timeout = 1200)]
public void CheckMethod(byte []bytes)
{
     PexAssume.IsNotNull(bytes);
     ImplCheckData.Crc32.Puzzle(bytes);
}
```

Как выглядят результаты

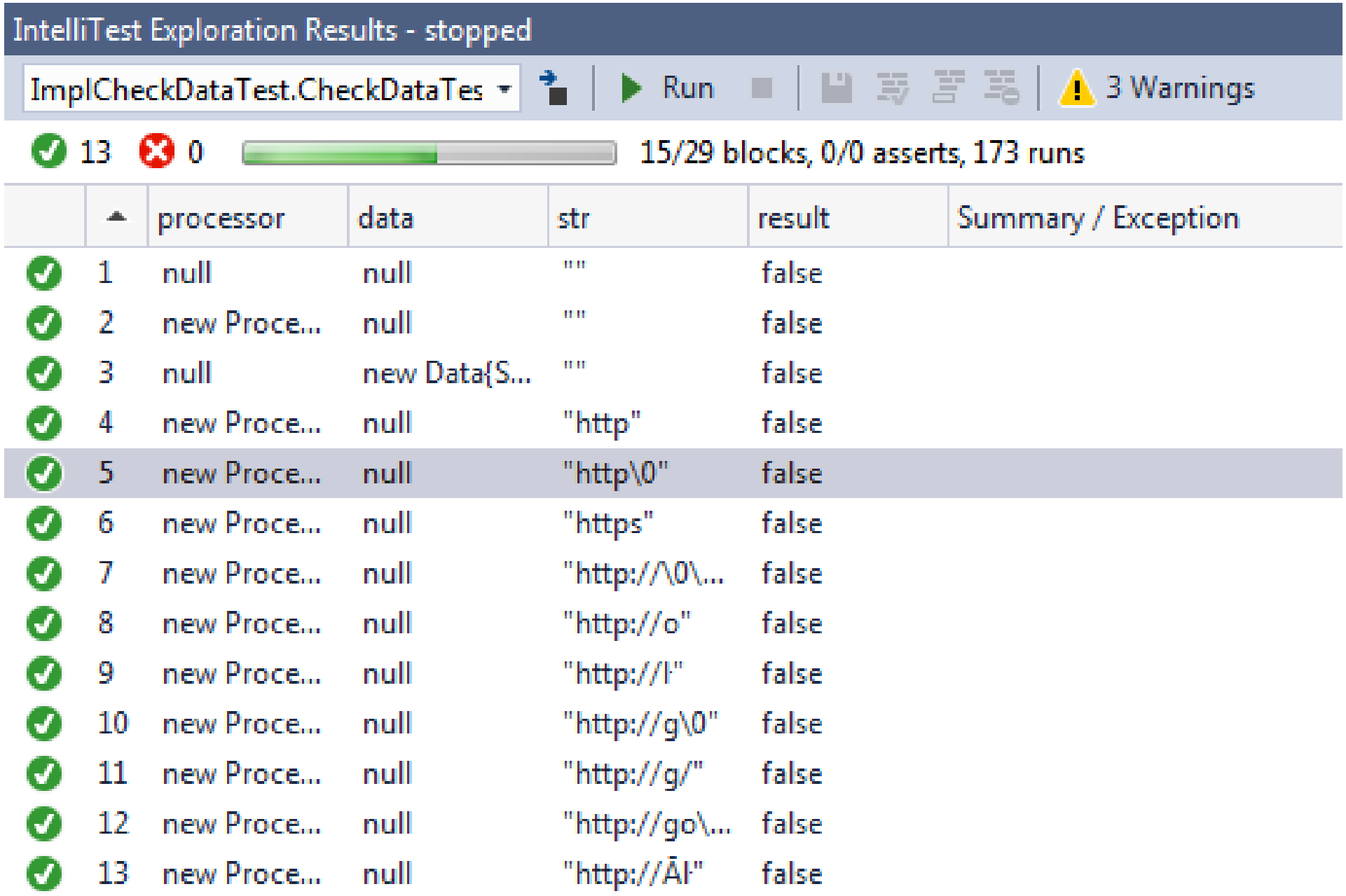


Пример для WhiteBox тестирования

```
public static bool CheckData(Processor processor, Data data, string url)
{
    if (processor == null && data == null)
        return false;

    var regex = new Regex(UrlRegex, RegexOptions.IgnoreCase);
    var result = regex.Match(url);

    if (result.Success && result.Groups[2].Value == "microsoft.com")
        return processor.Foo(data);

    return false;
}
```

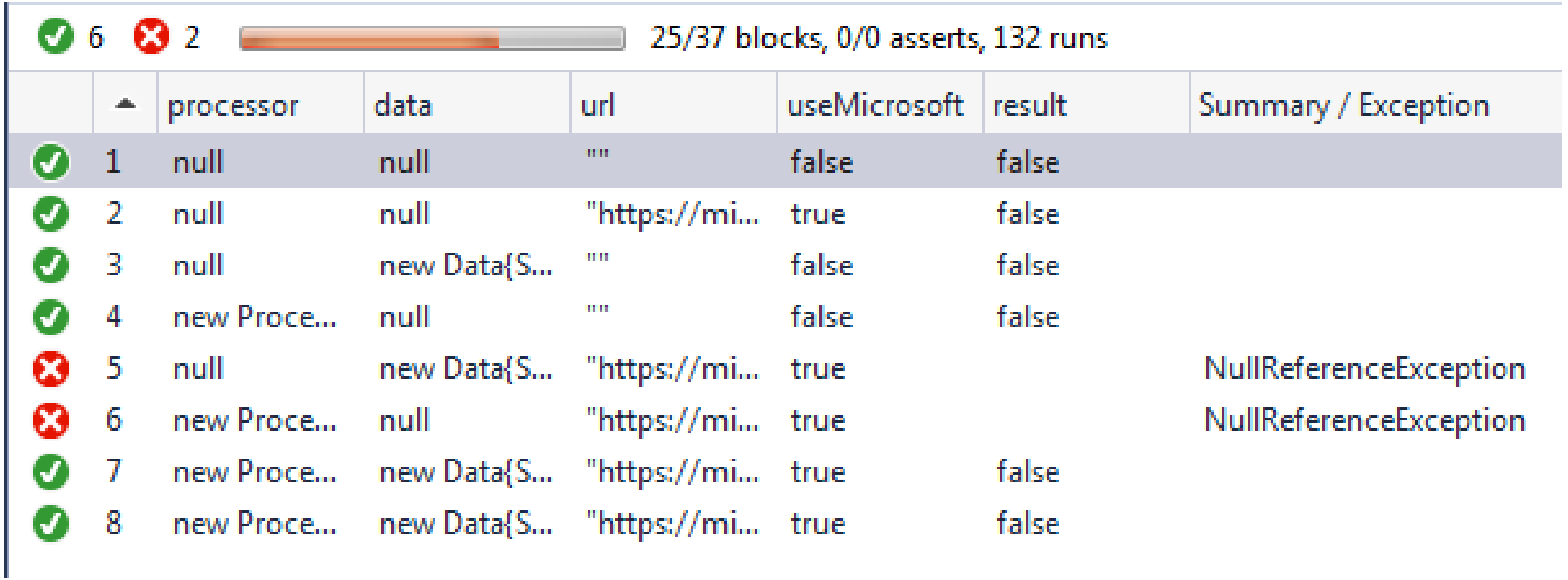
Как выглядит PEX-тест

```
[PexMethod]
public bool CheckDataTest(Processor processor,
                Data data,
                string url) {

   PexAssume.IsNotNull(url);
   bool result = ImplCheckData.CheckData(processor, data, url);
   return result;
}
```


IntelliTest Exploration Results - stopped
ImplCheckDataTest.CheckDataTes
Run
3 Warnings
13
0
15/29 blocks, 0/0 asserts, 173 runs
processor
data
str
result
Summary / Exception
1 null null "" false
2 new Proce... null "" false
3 null new Data{S... "" false
4 new Proce... null "http" false
5 new Proce... null "http\0" false
6 new Proce... null "https" false
7 new Proce... null "http://\0\... false
8 new Proce... null "http://o" false
9 new Proce... null "http://ł" false
10 new Proce... null "http://g\0" false
11 new Proce... null "http://g/" false
12 new Proce... null "http://go\... false
13 new Proce... null "http://Āł" false

IntelliTest - это не одна кнопка «сделай хорошо»

```
[PexMethod]
public bool CheckDataTest(Processor processor,
        Data data,
        string url,
        bool useMicrosoft
        ) {
    PexAssume.IsNotNull(url);
    PexAssume.IsNotNull(!useMicrosfot || url == “https://microsoft.com");
    bool result = ImplCheckData.CheckData(processor, data, url);
    return result;
}
```



6 2 25/37 blocks, 0/0 asserts, 132 runs

	processor	data	url	useMicrosoft	result	Summary / Exception
1	null	null	""	false	false	
2	null	null	"https://mi...	true	false	
3	null	new Data{S...	""	false	false	
4	new Proce...	null	""	false	false	
5	null	new Data{S...	"https://mi...	true		NullReferenceException
6	new Proce...	null	"https://mi...	true		NullReferenceException
7	new Proce...	new Data{S...	"https://mi...	true	false	
8	new Proce...	new Data{S...	"https://mi...	true	false	

Статическое символьное выполнение

- Идея: выкинем обычную виртуальную машину, оставим только символьную

Статическое символьное выполнение

- Идея: выкинем обычную виртуальную машину, оставим только символьную

- Будем тестировать каждую функцию в проекте отдельно, при этом не будем требовать корректность символьного выполнения

Статическое символьное выполнение

- Идея: выкинем обычную виртуальную машину, оставим только символьную
- Будем тестировать каждую функцию в проекте отдельно, при этом не будем требовать корректность символьного выполнения
- В результате можно добиться полного анализа больших проектов за несколько десятков минут

Основные отличия от поиска ошибочных шаблонов

- Производится межпроцедурный анализ (причина ошибки и место её возникновения могут находится в разных методах)

Основные отличия от поиска ошибочных шаблонов

- Производится «межпроцедурный» анализ (причина ошибки и место её возникновения могут находится в разных методах)

- Используются SMT-Решатели для отсеивания невозможных ситуаций (ложные срабатывания)

Основные отличия от поиска ошибочных шаблонов

- Производится «межпроцедурный» анализ (причина ошибки и место её возникновения могут находится в разных методах)
- Используются SMT-Решатели для отсеивания невозможных ситуаций (ложные срабатывания)
- Для описания ошибок используются трассы, т.е. упорядоченные наборы точек в программе

Пример сложной ошибки

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;
    if (args.Length > 0)
       foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes")
       verbose = true;
   Log(foo, verbose);
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;
    if (args.Length > 0)
       foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes")
       verbose = true;
   Log(foo, verbose);    // Possible NullReferenceException
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false; // (1) foo is assigned to null
    if (args.Length > 0)
       foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes")
       verbose = true;
   Log(foo, verbose);
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;  // (1) foo is assigned to null
    if (args.Length > 0) // (2) taking false branch
        foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes")
        verbose = true;
   Log(foo, verbose);
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;  // (1) foo is assigned to null
    if (args.Length > 0) // (2) taking false branch
        foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes") // (3) taking true branch
        verbose = true;
   Log(foo, verbose);
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
                Console.WriteLine(foo.Message);
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;  // (1) foo is assigned to null
    if (args.Length > 0) // (2) taking false branch
       foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes") // (3) taking true branch
       verbose = true;
   Log(foo, verbose); // (4) NRE occurs inside method Log
}
```

Пример трассы

```
static void Log(Foo foo, bool verbose) {
            if (verbose)
               Console.WriteLine(foo.Message); //(5) NRE occurs here
}

static void Main(string[] args) {
    Foo foo = null; bool verbose = false;  // (1) foo is assigned to null
    if (args.Length > 0) // (2) taking false branch
       foo = GetFoo();
    if (Environment.GetEnvironmentVariable("VERBOSE") == "yes") // (3) taking true branch
       verbose = true;
   Log(foo, verbose); // (4) NRE occurs inside method Log
}
```

Инструменты поиска сложных ошибок

- SharpChecker, ISP RAS
- Coverity, Synopsys
- Klocwork, Rogue Wave Software
- dotTest, Parasoft

SharpChecker

- Символьное выполнение для поиска сложных ошибок, включая межпроцедурный анализ

- Бесплатная пробная версия, которую можно запустить на вашем закрытом коде

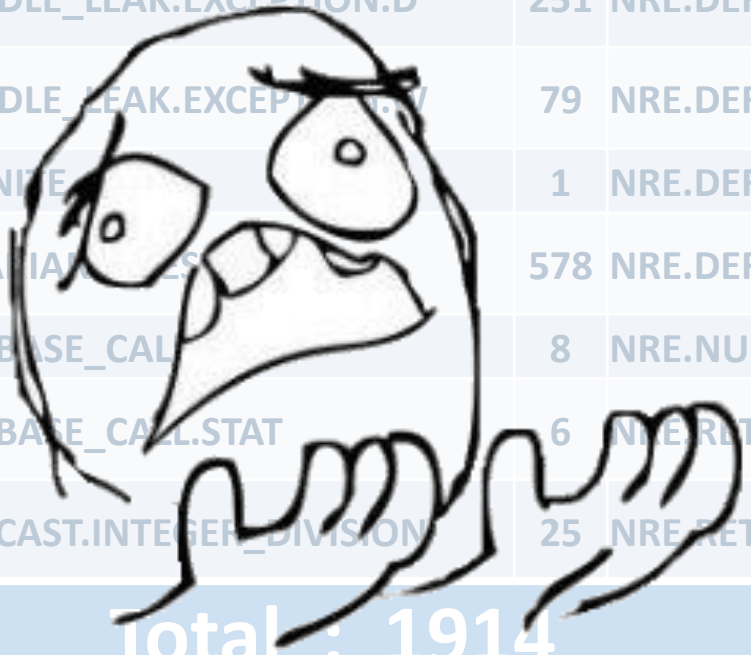
Часть 3. Интеграция в процесс разработки

Типичное первое использование

Warning	#	Warning	#	Warning	#
BAD_COPY_PASTE	2	NO_CAST.INTEGER_OVERFLOW	95	NRE.RET.USER.PROC	3
CAST_AFTER_CHECK	1	NO_LOCK.STAT	107	NRE.RET.USER.PROC.ARGUMENT	3
HANDLE_LEAK	78	NRE	7	NRE.RET.USER.PROC.STATISTICAL	30
HANDLE_LEAK.EXCEPTION	401	NRE.ARGUMENT	20	SIMILAR_BRANCHES	20
HANDLE_LEAK.EXCEPTION.D	251	NRE.DEREF_AFTER_AS	4	UNREACHABLE_CODE	94
HANDLE_LEAK.EXCEPTION.W	79	NRE.DEREF_AFTER_AS.INSTANT	1	UNREACHABLE_CODE.EXCEPTION	9
INFINITE_LOOP	1	NRE.DEREF_AFTER_NULL	18	UNREACHABLE_CODE.SYNTAX	3
INVARIANT_RESULT	578	NRE.DEREF_AFTER_NULL.ARGUMENT	3	UNUSED_VALUE	22
NO_BASE_CALL.LIB	8	NRE.NULL_AFTER_DEREF	27	WRONG_ARGUMENTS_ORDER	1
NO_BASE_CALL.STAT	6	NRE.RET.LIB.PROC.Container.STATISTICAL	6	WRONG_SEMICOLON	1
NO_CAST.INTEGER_DIVISION	25	NRE.RET.LIB.PROC.STATISTICAL	10		
Total : 1914					

Типичное первое использование

Warning	#	Warning	#	Warning	#
BAD_COPY_PASTE	2	NO_CAST.INTEGER_OVERFLOW	95	NRE.RET.USER.PROC	3
CAST_AFTER_CHECK	1	NO_LOCK.STAT	107	NRE.RET.USER.PROC.ARGUMENT	3
HANDLE_LEAK	78	NRE	7	NRE.RET.USER.PROC.STATISTICAL	30
HANDLE_LEAK.EXCEPTION	401	NRE.ARGUMENT	20	SIMILAR_BRANCHES	20
HANDLE_LEAK.EXCEPTION.D	251	NRE.DEREF_AFTER_AS	4	UNREACHABLE_CODE	94
HANDLE_LEAK.EXCEPTION.V	79	NRE.DEREF_AFTER_AS.INSTANT	1	UNREACHABLE_CODE.EXCEPTION	9
INFINITE	1	NRE.DEREF_AFTER_NULL	18	UNREACHABLE_CODE.SYNTAX	3
INVARIANT_CLASS	578	NRE.DEREF_AFTER_NULL.ARGUMENT	3	UNUSED_VALUE	22
NO_BASE_CALL	8	NRE.NULL_AFTER_DEREF	27	WRONG_ARGUMENTS_ORDER	1
NO_BASE_CALL.STAT	6	NRE.RET.LIB.PROC.Container.STATISTICAL	6	WRONG_SEMICOLON	1
NO_CAST.INTEGER_DIVISION	25	NRE.RET.LIB.PROC.STATISTICAL	10		
Total : 1914					

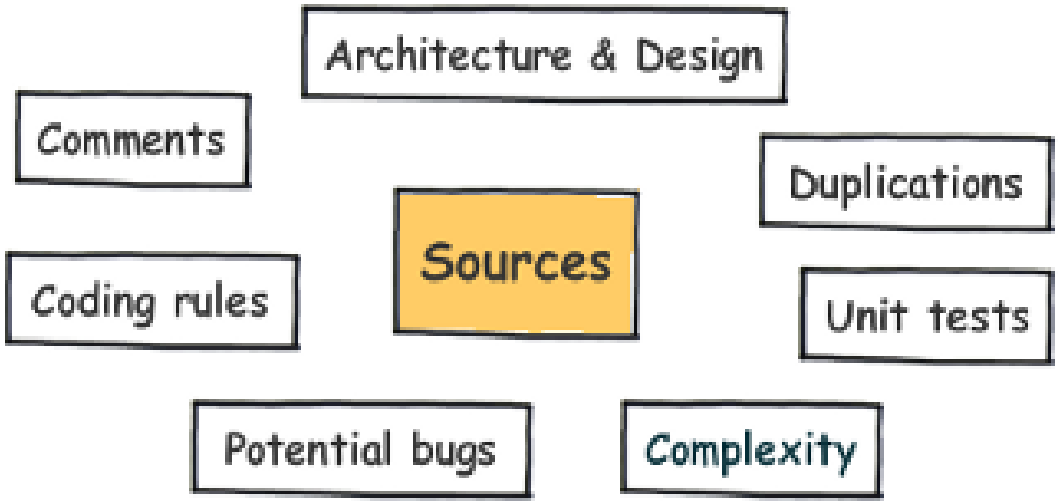


Куда встраивать статический анализ

- В IDE, показывать результаты анализа прямо во время написания кода
- После локальной сборки
- В систему code review
- После ночной сборки

SonarQube

- Поддержка различных анализаторов
- Интеграция с системами сборки
- Интеграция с баг-трекерами
- Удобный веб-интерфейс
- https://nemo.sonarqube.com



Вопросы

vedun@ispras.ru

telegram: @vkkoshelev

http://sharpchecker.ispras.ru