Cooperative Testing and Analysis: Human-Tool, Tool-Tool, and Human-Human Cooperations to Get Work Done

Tao Xie
Peking University, China (2011-2012)
North Carolina State University, USA

Supported in part by NSF CAREER 0845272 and Microsoft Research SEIF 2011

Why Automate Testing?

- Software testing is **important**
  - Software errors cost the U.S. economy about $59.5 billion each year (0.6% of the GDP) [NIST 02]
  - Improving testing infrastructure could save 1/3 cost [NIST 02]
- Software testing is **costly**
  - Account for even half the total cost of software development [Beizer 90]
- Automated testing **reduces** manual testing **effort**
  - Test execution: Junit/xUnit framework
  - Test generation: AgitarOne, Parasoft Jtest, etc.
  - Test-behavior checking: AgitarOne, Parasoft Jtest, etc.

Dynamic Symbolic Execution

[Godefroid et al. 05]

Choose next path

Solve

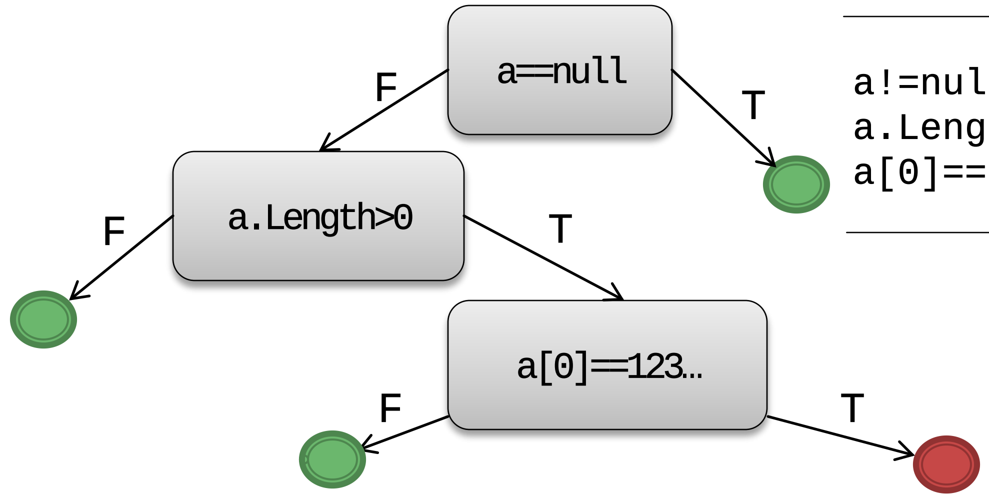
Execute&Monitor

Code to generate inputs for:

```
void CoverMe(int[] a)
{
  if (a == null) return;
  if (a.Length > 0)
    if (a[0] == 1234567890)
      throw new Exception("bug");
}
```

Constraints to solve	Data	Observed constraints
	null	a==null
a!=null	{}	a!=null && !(a.Length>0)
a!=null && a.Length>0	[illegible]	[illegible]
a!=null && a.Length>0 && a[0]==1234567890	{123..}	a!=null && a.Length>0 && a[0]==1234567890

Negated condition



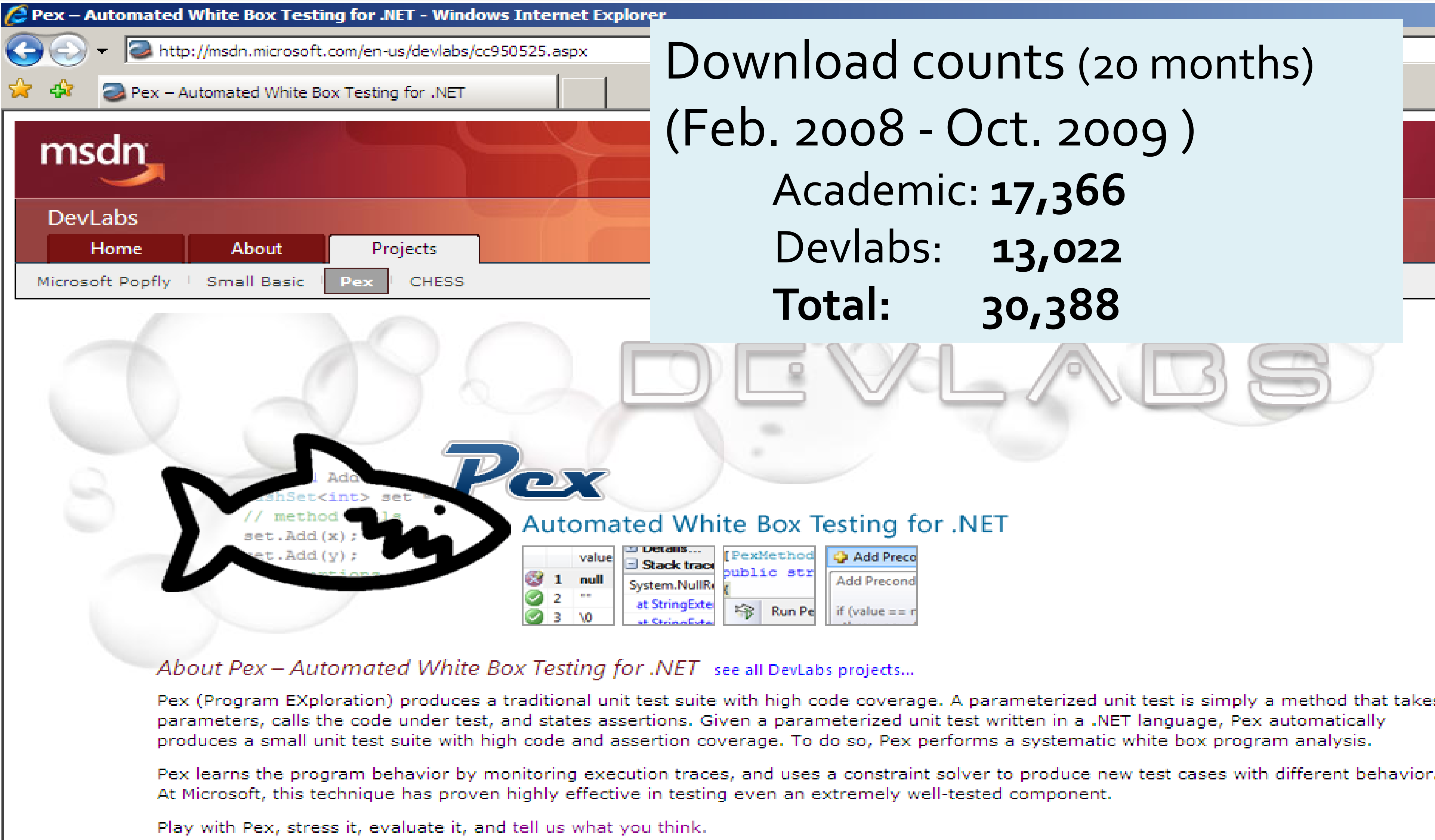
Done: There is no path left.

Automating Test Generation

@NCSU ASE

- Method sequences
  - MSeqGen/Seeker [Thummalapenta et al. OOSPLA 11, ESEC/FSE 09], Covana [Xiao et al. ICSE 2011], OCAT [Jaygarl et al. ISSTA 10], Evacon [Inkumsah et al. ASE 08], Symclat [d'Amorim et al. ASE 06]
- Environments e.g., db, file systems, network, …
  - DBApp Testing [Taneja et al. ESEC/FSE 11], [Pan et al. ASE 11]
  - CloudApp Testing [Zhang et al. IEEE Soft 12]
- Loops
  - **Fitnex [Xie et al. DSN 09]**
- Code evolution
  - eXpress [Taneja et al. ISSTA 11]

Pex on MSDN DevLabs
Incubation Project for Visual Studio


Pex – Automated White Box Testing for .NET - Windows Internet Explorer
http://msdn.microsoft.com/en-us/devlabs/cc950525.aspx
Pex – Automated White Box Testing for .NET
msdn
DevLabs
Home
About
Projects
Microsoft Popfly
Small Basic
Pex
CHESS
DEVLABS
Pex
Automated White Box Testing for .NET

About Pex – Automated White Box Testing for .NET see all DevLabs projects...

Pex (Program EXploration) produces a traditional unit test suite with high code coverage. A parameterized unit test is simply a method that take parameters, calls the code under test, and states assertions. Given a parameterized unit test written in a .NET language, Pex automatically produces a small unit test suite with high code and assertion coverage. To do so, Pex performs a systematic white box program analysis.

Pex learns the program behavior by monitoring execution traces, and uses a constraint solver to produce new test cases with different behavior At Microsoft, this technique has proven highly effective in testing even an extremely well-tested component.

Play with Pex, stress it, evaluate it, and tell us what you think.

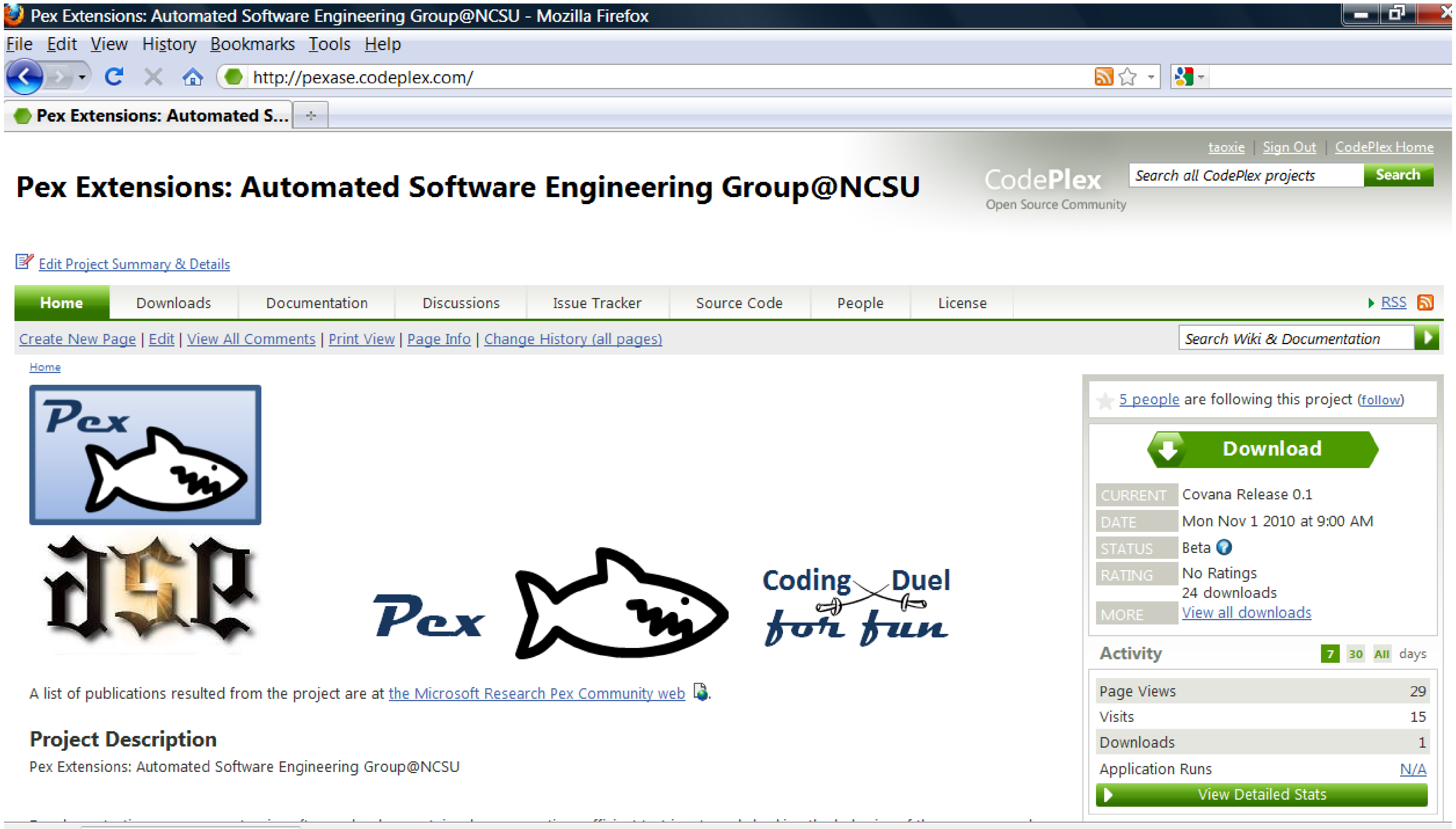
Download counts (20 months)
(Feb. 2008 - Oct. 2009)

Academic:	**17,366**
Devlabs:	**13,022**
Total:	**30,388**

Open Source Pex extensions

http://pexase.codeplex.com/

Publications: http://research.microsoft.com/en-us/projects/pex/community.aspx#publications



Pex Extensions: Automated Software Engineering Group@NCSU - Mozilla Firefox

http://pexase.codeplex.com/

taoxie | Sign Out | CodePlex Home

Pex Extensions: Automated Software Engineering Group@NCSU

Edit Project Summary & Details

Home | Downloads | Documentation | Discussions | Issue Tracker | Source Code | People | License

Create New Page | Edit | View All Comments | Print View | Page Info | Change History (all pages)

A list of publications resulted from the project are at the Microsoft Research Pex Community web.

Project Description

Pex Extensions: Automated Software Engineering Group@NCSU

5 people are following this project (follow)

Download

CURRENT	Covana Release 0.1
DATE	Mon Nov 1 2010 at 9:00 AM
STATUS	Beta
RATING	No Ratings 24 downloads
MORE	View all downloads

Activity 7 30 All days

Page Views	29
Visits	15
Downloads	1
Application Runs	N/A

View Detailed Stats

Reality Check

- **Machine is better at task set A**
  - Mechanical, tedious, repetitive tasks, …
  - Ex. solving constraints along a long path

- **Human is better at task set B**
  - Intelligence, human intention, abstraction, domain knowledge, …
  - Ex. local reasoning after a loop

= A U B?

Dagstuhl Seminar 10111

Practical Software Testing: **Tool Automation** and **Human Factors**

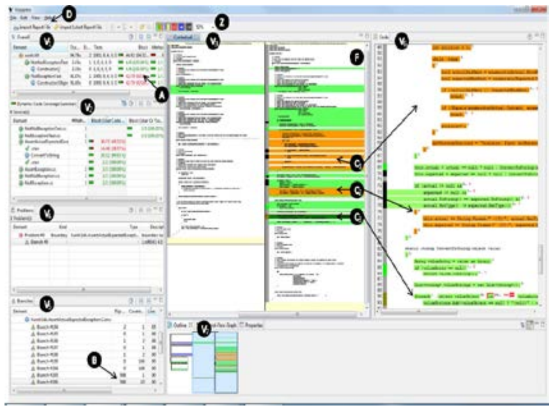
Human Factors

Dagstuhl Seminar 10111

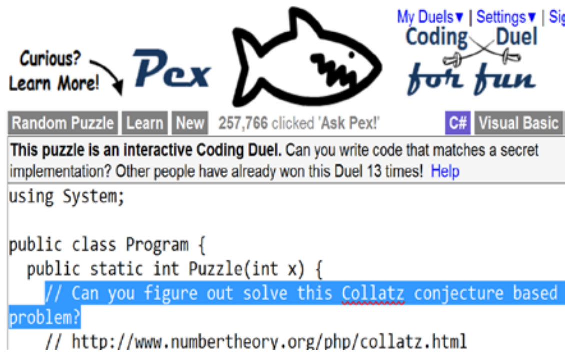
Practical Software Testing: **Tool Automation** and Human Factors

Cooperation Between Human and Machine

- Human-<u>Assisted</u> Computing
  - Driver: tool←→ Helper: human
  - Ex. Covana [Xiao et al. ICSE 2011]



- Human-<u>Centric</u> Computing
  - Driver: human ←→ Helper: tool
  - Ex. Coding duels @Pex for Fun



This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 13 times! Help

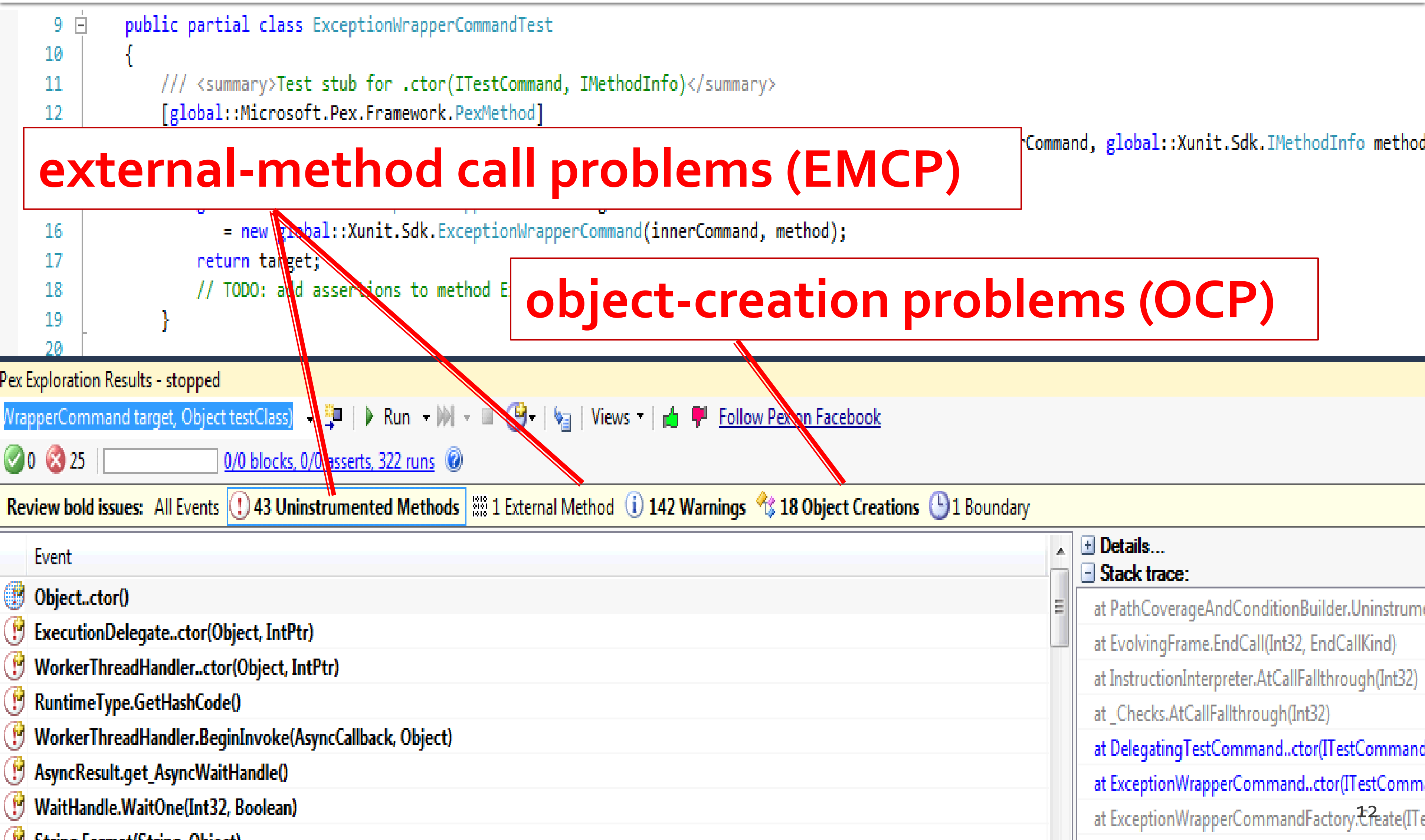
```
using System;

public class Program {
  public static int Puzzle(int x) {
    // Can you figure out solve this Collatz conjecture based
problem?
    // http://www.numbertheory.org/php/collatz.html
```

***Interfaces** are important. **Contents** are important too!*

Human-Assisted Computing

- Motivation
  - Tools are often not powerful enough (at least for now)
  - Human is good at some aspects that tools are not
- **Task for Tool**: What needs to automate?
- Tool → Human
  - What difficulties does the tool face?
  - How to communicate info to the user to get her help?
- Tool ←Human
  - How does the user help the tool based on the info?
- Iterations to form feedback loop?

Problems Faced by Automated-Structural-Test-Generation Tools



```
public partial class ExceptionWrapperCommandTest
{
    /// <summary>Test stub for .ctor(ITestCommand, IMethodInfo)</summary>
    [global::Microsoft.Pex.Framework.PexMethod]
                                                    Command, global::Xunit.Sdk.IMethodInfo method

            = new global::Xunit.Sdk.ExceptionWrapperCommand(innerCommand, method);
        return target;
        // TODO: add assertions to method E
    }
```

external-method call problems (EMCP)

object-creation problems (OCP)

Pex Exploration Results - stopped

WrapperCommand target, Object testClass) Run Views Follow Pex on Facebook

0 25 0/0 blocks, 0/0 asserts, 322 runs

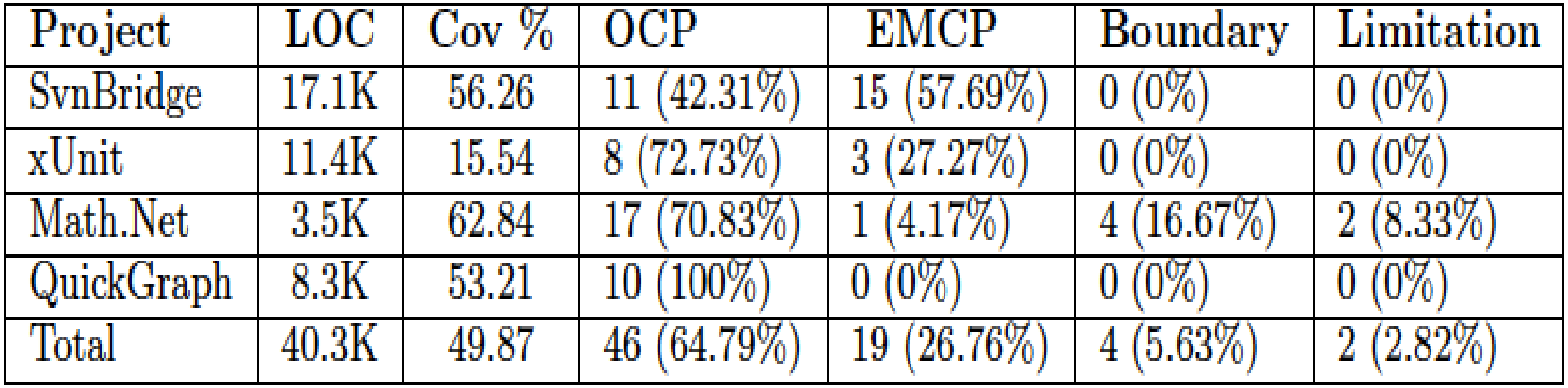
Review bold issues: All Events 43 Uninstrumented Methods 1 External Method 142 Warnings 18 Object Creations 1 Boundary

```
Event
Object..ctor()
ExecutionDelegate..ctor(Object, IntPtr)
WorkerThreadHandler..ctor(Object, IntPtr)
RuntimeType.GetHashCode()
WorkerThreadHandler.BeginInvoke(AsyncCallback, Object)
AsyncResult.get_AsyncWaitHandle()
WaitHandle.WaitOne(Int32, Boolean)
String.Format(String, Object)
```

Details...
Stack trace:

```
at PathCoverageAndConditionBuilder.Uninstrum
at EvolvingFrame.EndCall(Int32, EndCallKind)
at InstructionInterpreter.AtCallFallthrough(Int32)
at _Checks.AtCallFallthrough(Int32)
at DelegatingTestCommand..ctor(ITestCommand
at ExceptionWrapperCommand..ctor(ITestComm
at ExceptionWrapperCommandFactory.Create(ITe
```

DSE Challenges - Preliminary Study



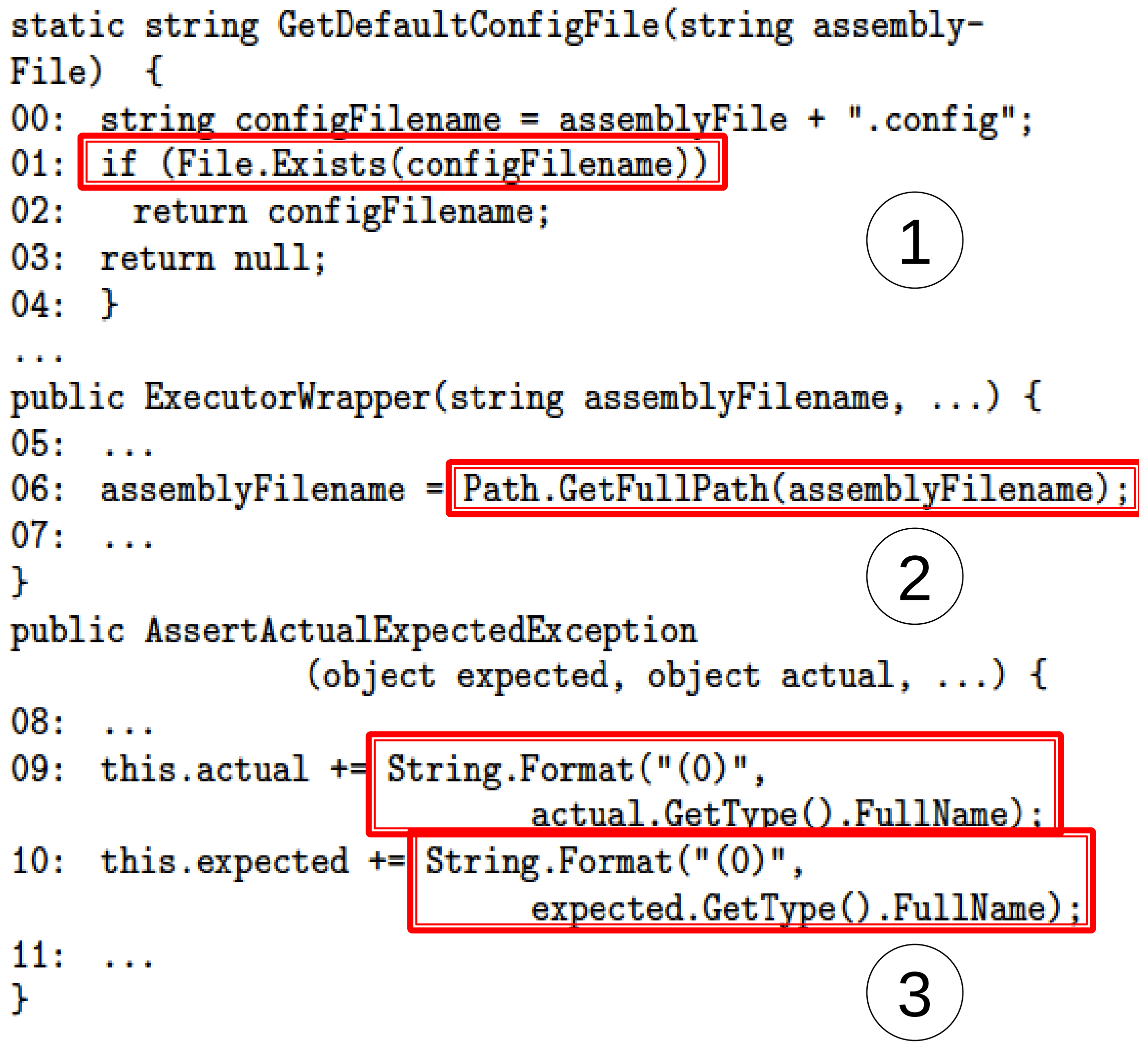
Project	LOC	Cov %	OCP	EMCP	Boundary	Limitation
SvnBridge	17.1K	56.26	11 (42.31%)	15 (57.69%)	0 (0%)	0 (0%)
xUnit	11.4K	15.54	8 (72.73%)	3 (27.27%)	0 (0%)	0 (0%)
Math.Net	3.5K	62.84	17 (70.83%)	1 (4.17%)	4 (16.67%)	2 (8.33%)
QuickGraph	8.3K	53.21	10 (100%)	0 (0%)	0 (0%)	0 (0%)
Total	40.3K	49.87	46 (64.79%)	19 (26.76%)	4 (5.63%)	2 (2.82%)

The total block coverage achieved is 49.87%, with the lowest coverage being 15.54%.

- object-creation problems (OCP) - 64.79%
- external-method call problems (EMCP) - 26.76%
- boundary problems – 5.63%
- limitations of the used constraint solver – 2.82%

External-Method Call Problems (EMCP) Example

- Example 1:
  - **File.Exists** has data dependencies on program input
  - Subsequent branch at Line 1 using the return value of **File.Exists**.
- Example 2:
  - **Path.GetFullPath** has data dependencies on program input
  - **Path.GetFullPath** throws exceptions.
- Example 3: **Stirng.Format** do not cause any problem



```
static string GetDefaultConfigFile(string assembly-
File)  {
00:  string configFilename = assemblyFile + ".config";
01:  if (File.Exists(configFilename))
02:    return configFilename;
03:  return null;
04:  }
...
public ExecutorWrapper(string assemblyFilename, ...) {
05:  ...
06:  assemblyFilename = Path.GetFullPath(assemblyFilename);
07:  ...
}
public AssertActualExpectedException
                       (object expected, object actual, ...) {
08:  ...
09:  this.actual += String.Format("(0)",
                       actual.GetType().FullName);
10:  this.expected += String.Format("(0)",
                       expected.GetType().FullName);
11:  ...
}
```

Figure 1: Three simplified methods from xUnit

Cooperation Between Human and Test-Generation Tools

- Motivation
  - Tools are often not powerful enough (at least for now)
    - EMCPs and OCPs
  - Human is good at some aspects that tools are not
    - EMCPs: Instruct which external methods
      - to instrument
      - to write mock objects for
    - OCPs: Write factory methods for generating objects

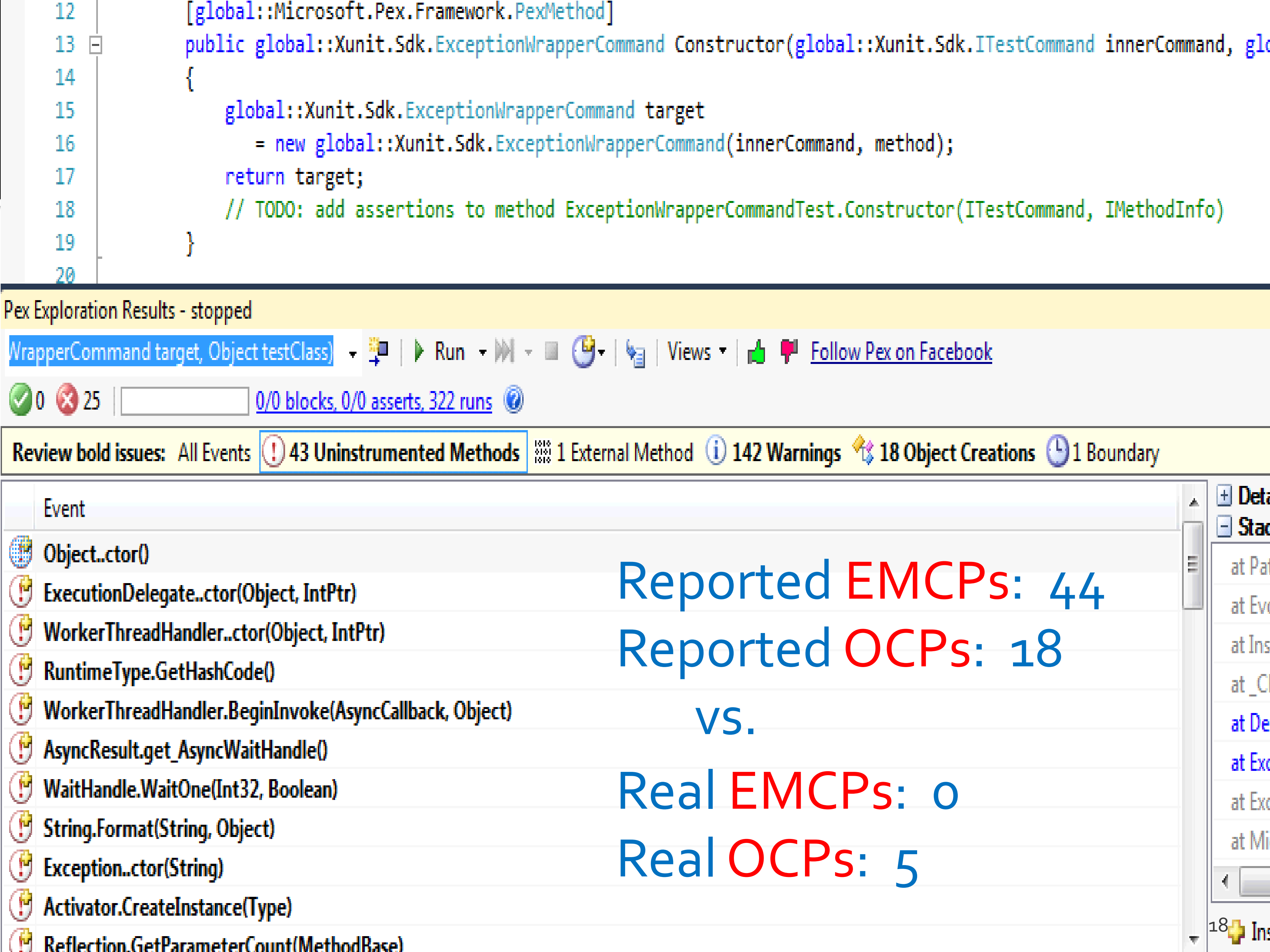
Cooperative Developer Testing

- Developers provide guidance to help tools achieve higher structural coverage

  - Apply tools to generate tests
  - Tools report achieved coverage & problems
  - Developers provide guidance
    - EMCP: Instrumentation or Mock Objects
    - OCP: Factory Methods

Existing Solution of Problem Identification

- Existing solution
  - identify all executed external-method calls
  - report all the non-primitive object types of program inputs and their fields
- Limitations
  - the number could be high
  - some identified problem are irrelevant for achieving higher structural coverage



```
[global::Microsoft.Pex.Framework.PexMethod]
public global::Xunit.Sdk.ExceptionWrapperCommand Constructor(global::Xunit.Sdk.ITestCommand innerCommand, glo
{
    global::Xunit.Sdk.ExceptionWrapperCommand target
       = new global::Xunit.Sdk.ExceptionWrapperCommand(innerCommand, method);
    return target;
    // TODO: add assertions to method ExceptionWrapperCommandTest.Constructor(ITestCommand, IMethodInfo)
}
```

Pex Exploration Results - stopped

WrapperCommand target, Object testClass) Run Views Follow Pex on Facebook

0 25 0/0 blocks, 0/0 asserts, 322 runs

Review bold issues: All Events 43 Uninstrumented Methods 1 External Method 142 Warnings 18 Object Creations 1 Boundary

Event
- Object..ctor()
- ExecutionDelegate..ctor(Object, IntPtr)
- WorkerThreadHandler..ctor(Object, IntPtr)
- RuntimeType.GetHashCode()
- WorkerThreadHandler.BeginInvoke(AsyncCallback, Object)
- AsyncResult.get_AsyncWaitHandle()
- WaitHandle.WaitOne(Int32, Boolean)
- String.Format(String, Object)
- Exception..ctor(String)
- Activator.CreateInstance(Type)
- Reflection.GetParameterCount(MethodBase)

Reported EMCPs: 44

Reported OCPs: 18

vs.

Real EMCPs: 0

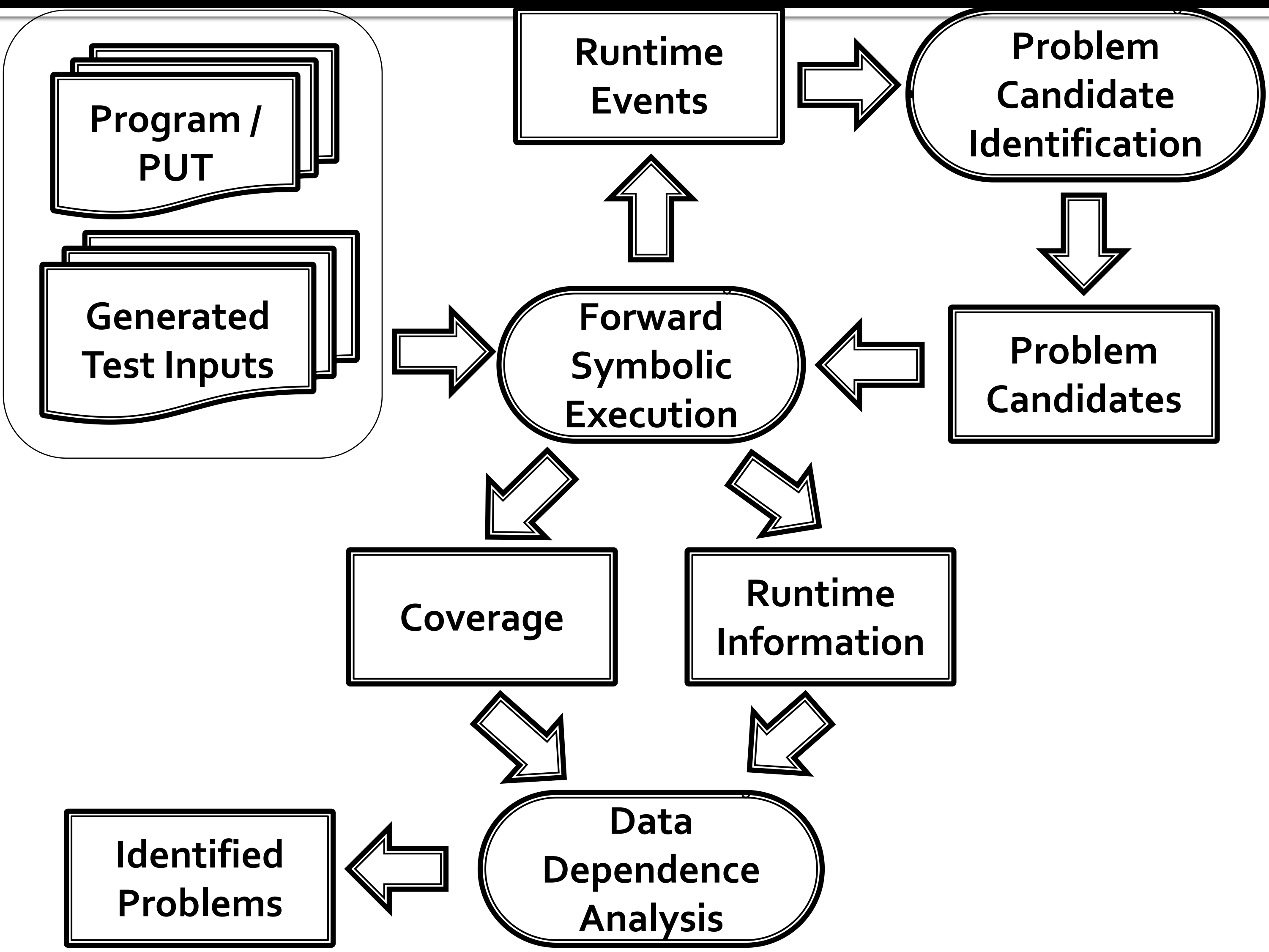
Real OCPs: 5

Proposed Approach: Covana

[Xiao et al. ICSE 11]

- Precisely identify problems faced by tools when achieving structural coverage
- Insight
  - Partially-Covered Statements (i.e., statements containing not-covered branches) have data dependency on real problem candidates
- Three main steps:
  - Problem Candidate Identification
  - Forward Symbolic Execution
  - Data Dependence Analysis

Overview of Covana



Data Dependence Analysis

Symbolic Expression:
return(File.Exists) == true

Element of
EMCP Candidate:
return(File.Exists)

Branch Statement Line 1 has data dependency on *File.Exists* at Line 1

```
static bool ParseCommandLine(string[] args,
                             out string assemblyFile, ...) {
00:  assemblyFile = args[0];
...
01:  if (!File.Exists(assemblyFile)) {
02:    Console.WriteLine("error: assem-
bly file not found: {0}", assemblyFile);
03:    return false;
04:  }
...
public Executor(string assemblyFilename) {
05:  this.assemblyFilename = Path.GetFullPath(assemblyFilename);
06:  ...
}
```

Evaluation – Subjects and Setup

- Subjects:
  - xUnit: unit testing framework for .NET
    - 223 classes and interfaces with 11.4 KLOC
  - QuickGraph: C# graph library
    - 165 classes and interfaces with 8.3 KLOC
- Evaluation setup:
  - Pex (0.24.50222.1) with the implemented extension as our DSE test-generation tool
  - Apply Pex to generate tests for program under test
  - Collect coverage and runtime information for identifying EMCPs and OCPs

Evaluation – Research Questions

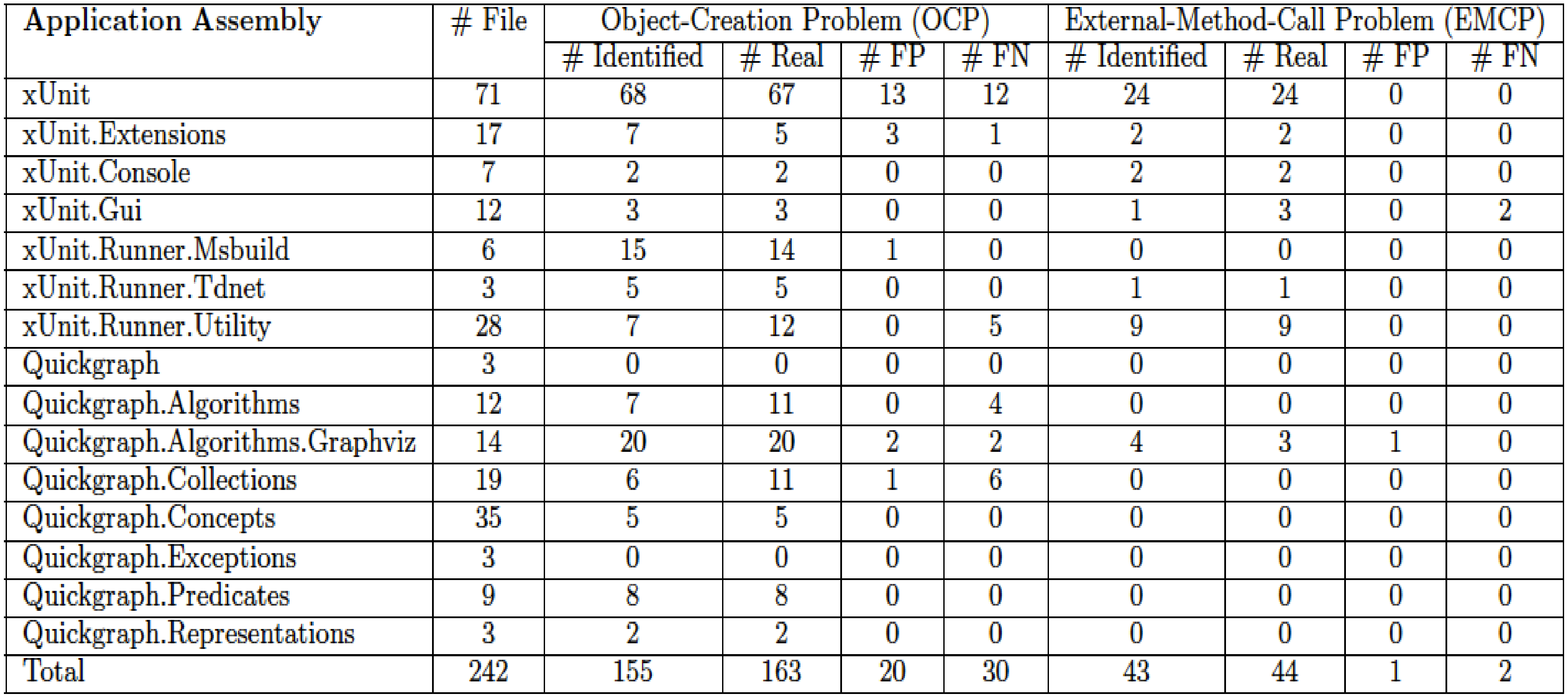
- RQ1: How effective is Covana in **identifying** the two main types of problems, EMCPs and OCPs?

- RQ2: How effective is Covana in **pruning** irrelevant problem candidates of EMCPs and OCPs?

Evaluations - RQ1: Problem Identification

Covana identifies

- 43 EMCPs with only 1 false positive and 2 false negatives
- 155 OCPs with 20 false positives and 30 false negatives.



Application Assembly	# File	Object-Creation Problem (OCP)				External-Method-Call Problem (EMCP)			
		# Identified	# Real	# FP	# FN	# Identified	# Real	# FP	# FN
xUnit	71	68	67	13	12	24	24	0	0
xUnit.Extensions	17	7	5	3	1	2	2	0	0
xUnit.Console	7	2	2	0	0	2	2	0	0
xUnit.Gui	12	3	3	0	0	1	3	0	2
xUnit.Runner.Msbuild	6	15	14	1	0	0	0	0	0
xUnit.Runner.Tdnet	3	5	5	0	0	1	1	0	0
xUnit.Runner.Utility	28	7	12	0	5	9	9	0	0
Quickgraph	3	0	0	0	0	0	0	0	0
Quickgraph.Algorithms	12	7	11	0	4	0	0	0	0
Quickgraph.Algorithms.Graphviz	14	20	20	2	2	4	3	1	0
Quickgraph.Collections	19	6	11	1	6	0	0	0	0
Quickgraph.Concepts	35	5	5	0	0	0	0	0	0
Quickgraph.Exceptions	3	0	0	0	0	0	0	0	0
Quickgraph.Predicates	9	8	8	0	0	0	0	0	0
Quickgraph.Representations	3	2	2	0	0	0	0	0	0
Total	242	155	163	20	30	43	44	1	2

Example Identified EMCPs - 1

ParseCommandLine, Pex achieved 44/154 (28.57%),

Branch Statement Line 1 has data dependency on *File.Exists* at Line 1

False branch at Line 1 is not covered

File.Exists is reported

```
static bool ParseCommandLine(string[] args,
                    out string assemblyFile, ...) {
00:  assemblyFile = args[0];
...
01:  if (!File.Exists(assemblyFile)) {
02:    Console.WriteLine("error: assem-
bly file not found: {0}", assemblyFile);
03:    return false;
04:  }
...
public Executor(string assemblyFilename) {
05:  this.assemblyFilename = Path.GetFullPath(assemblyFilename);
06:  ...
}
```

Example Identified EMCPs - 2

```
static bool ParseCommandLine(string[] args,
                             out string assemblyFile, ...) {
00:  assemblyFile = args[0];
...
01:  if (!File.Exists(assemblyFile)) {
02:    Console.WriteLine("error: assem-
bly file not found: {0}", assemblyFile);
03:    return false;
04:  }
...
public Executor(string assemblyFilename) {
05:  this.assemblyFilename = Path.GetFullPath(assemblyFilename);
06:  ...
}
```

Executor, Pex achieved 2/5 (40%)

Path.GetFullPath throws exceptions for all executions

Code after Line 6 is not covered

Path.GetFullPath is reported

Evaluations – RQ2: Irrelevant-Problem-Candidate Pruning

Covana prunes

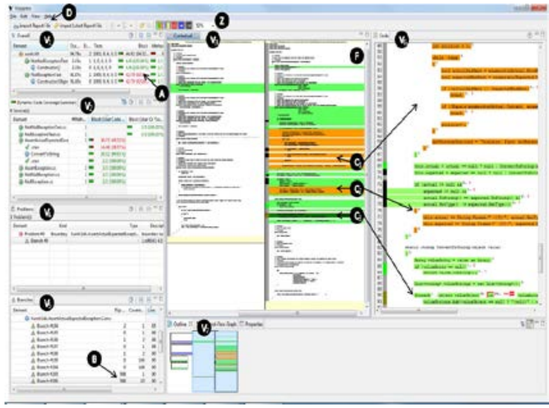
- 97.33% (1567 in 1610) EMCP candidates with 1 false positive and 2 false negatives
- 65.63% (296 in 451) OCP candidates with 20 false positives and 30 false negatives

Object-Creation Problem (OCP)				
#Candidate	#Identified	#Pruned	#FP	#FN
335	107	228 (68.06%)	17	18
116	48	68 (58.62%)	3	12
451	155	296 (65.63%)	20	30

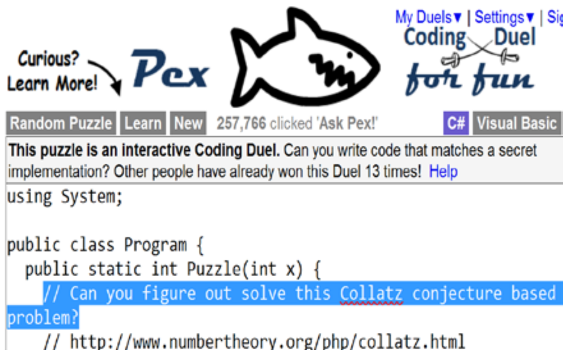
External-Method-Call Problem (EMCP)				
#Candidate	#Identified	#Pruned	#FP	#FN
1313	39	1274 (97.03%)	0	2
297	4	293 (98.65%)	1	0
1610	43	1567(97.33%)	1	2

Cooperation Between Human and Machine

- **Human**-**Assisted** **Computing**
  - Driver: tool←→ Helper: human
  - Ex. Covana [Xiao et al. ICSE 2011]



- **Human**-**Centric** **Computing**
  - Driver: human ←→ Helper: tool
  - Ex. Coding duels @Pex for Fun



This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 13 times! Help

```
using System;

public class Program {
  public static int Puzzle(int x) {
    // Can you figure out solve this Collatz conjecture based
problem?
    // http://www.numbertheory.org/php/collatz.html
```

***Interfaces** are important. **Contents** are important too!*

Behind the Scene of Pex for Fun

[ASE 08sp]

Secret Impl $\overset{\text{behavior}}{==}$ Player Impl?

Secret Implementation

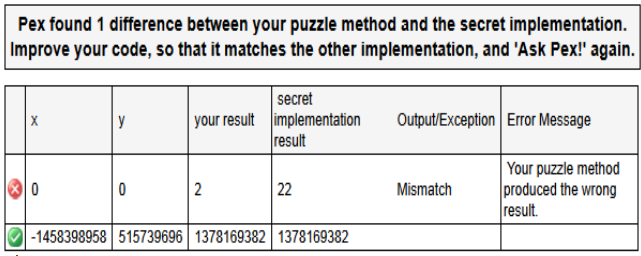
```
class Secret  {
    public static int Puzzle(int  x) {
        return x * 3 + 10;
    }
}
```

Player Implementation

```
class Player {
    public static int Puzzle(int  x) {
        return x;
    }
}
```

Ask Pex!

```
class Test {
    public static void Driver(int  x) {
        if (Secret.Puzzle(x) != Player.Puzzle(x))
            throw new Exception("Found a Difference");
    }
}
```

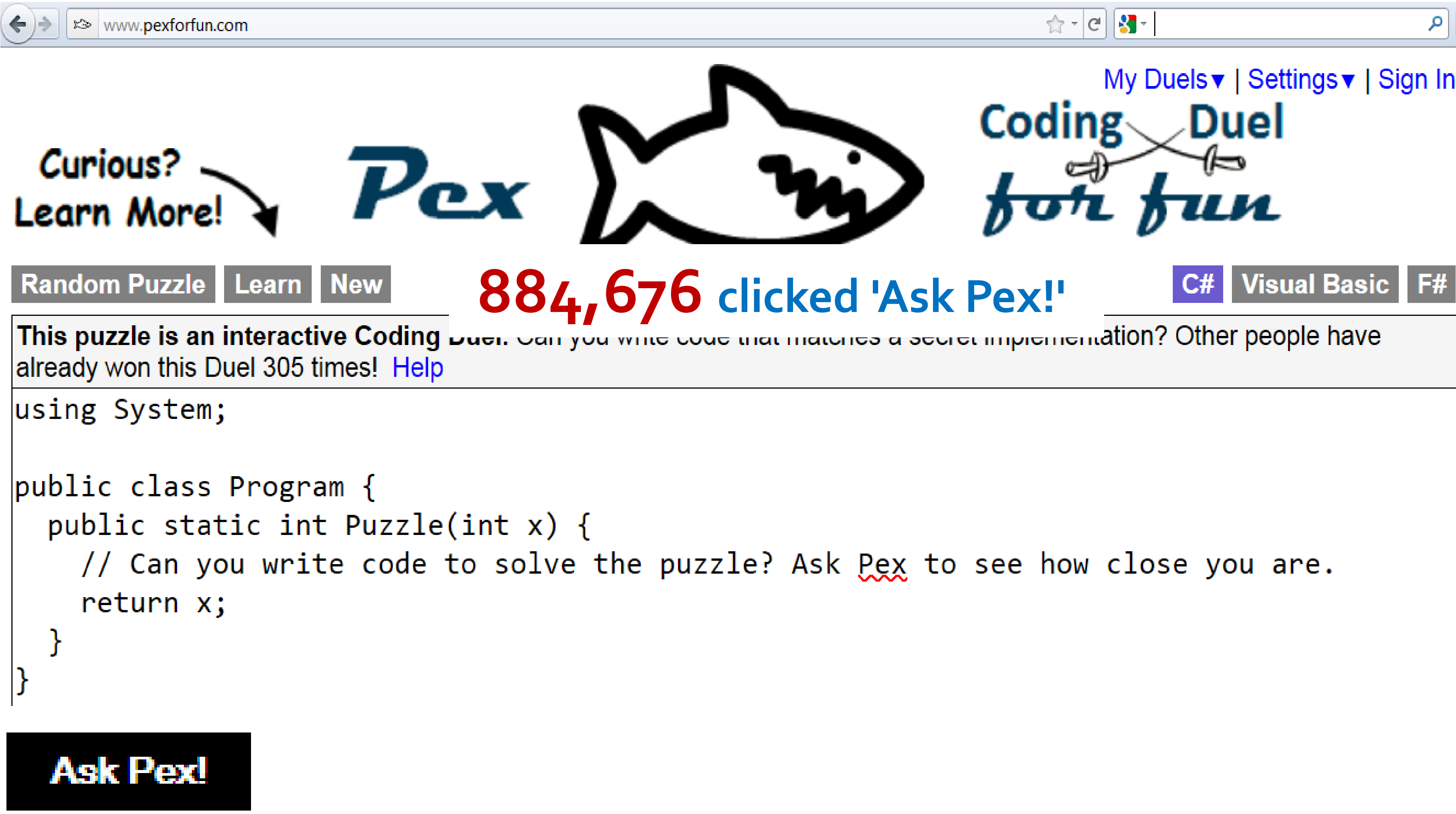


Pex found 1 difference between your puzzle method and the secret implementation. Improve your code, so that it matches the other implementation, and 'Ask Pex!' again.

	x	y	your result	secret implementation result	Output/Exception	Error Message
✗	0	0	2	22	Mismatch	Your puzzle method produced the wrong result.
✓	-1458398958	515739696	1378169382	1378169382		

Migrating Pex to the Web/Cloud

Try it at http://www.pexforfun.com/



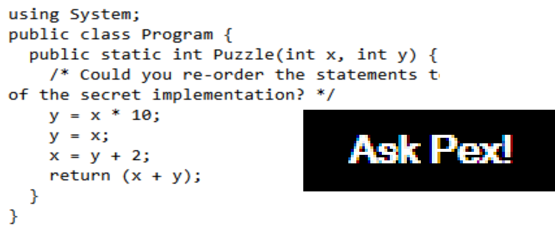
884,676 clicked 'Ask Pex!'

This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 305 times! Help

```
using System;

public class Program {
  public static int Puzzle(int x) {
    // Can you write code to solve the puzzle? Ask Pex to see how close you are.
    return x;
  }
}
```

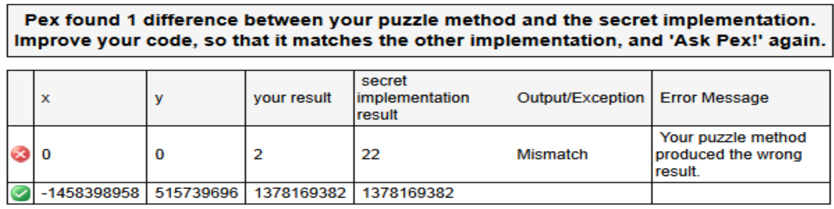
HCC: Pex for Fun (Human-Human C)

- Coding duels at http://www.pexforfun.com/
- **Task** for Human: write behavior-equiv code



```
using System;
public class Program {
  public static int Puzzle(int x, int y) {
    /* Could you re-order the statements t
of the secret implementation? */
    y = x * 10;
    y = x;
    x = y + 2;
    return (x + y);
  }
}
```

- Human → Tool
  - Does my new code behave differently? How exactly?



Pex found 1 difference between your puzzle method and the secret implementation. Improve your code, so that it matches the other implementation, and 'Ask Pex!' again.

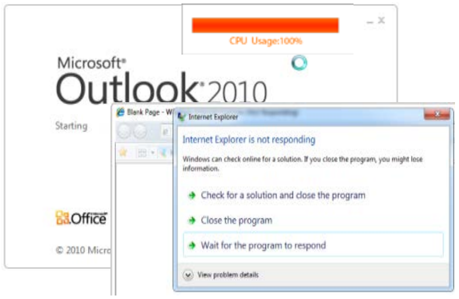
	x	y	your result	secret implementation result	Output/Exception	Error Message
✗	0	0	2	22	Mismatch	Your puzzle method produced the wrong result.
✓	-1458398958	515739696	1378169382	1378169382		

- Human ← Tool
  - Could you fix your code to handle **failed/passed tests**?

- Iterations to form feedback loop?
  - Yes, till tool generates no failed tests/player is impatient

Human-Human/Tool Cooperation: StackMine

[ICSE 12] in collaboration with MSR Asia



CPU Usage:100%

Microsoft® Outlook® 2010

Starting

Office

© 2010 Micro

Blank Page - W

Internet Explorer

Internet Explorer is not responding

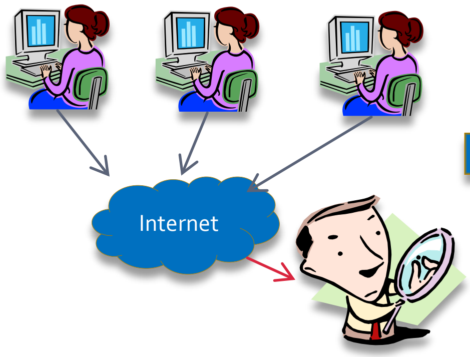
Windows can check online for a solution. If you close the program, you might lose information.

- Check for a solution and close the program
- Close the program
- Wait for the program to respond

View problem details

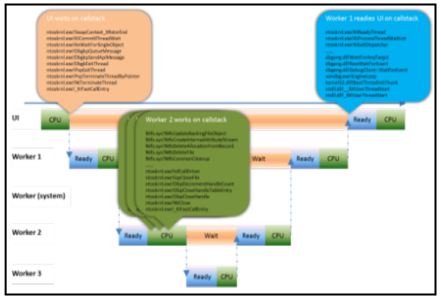
OS Performance in The Real World

- One of top user complaints
- Impacting large number of users every day
- High impact on usability and productivity



Challenges

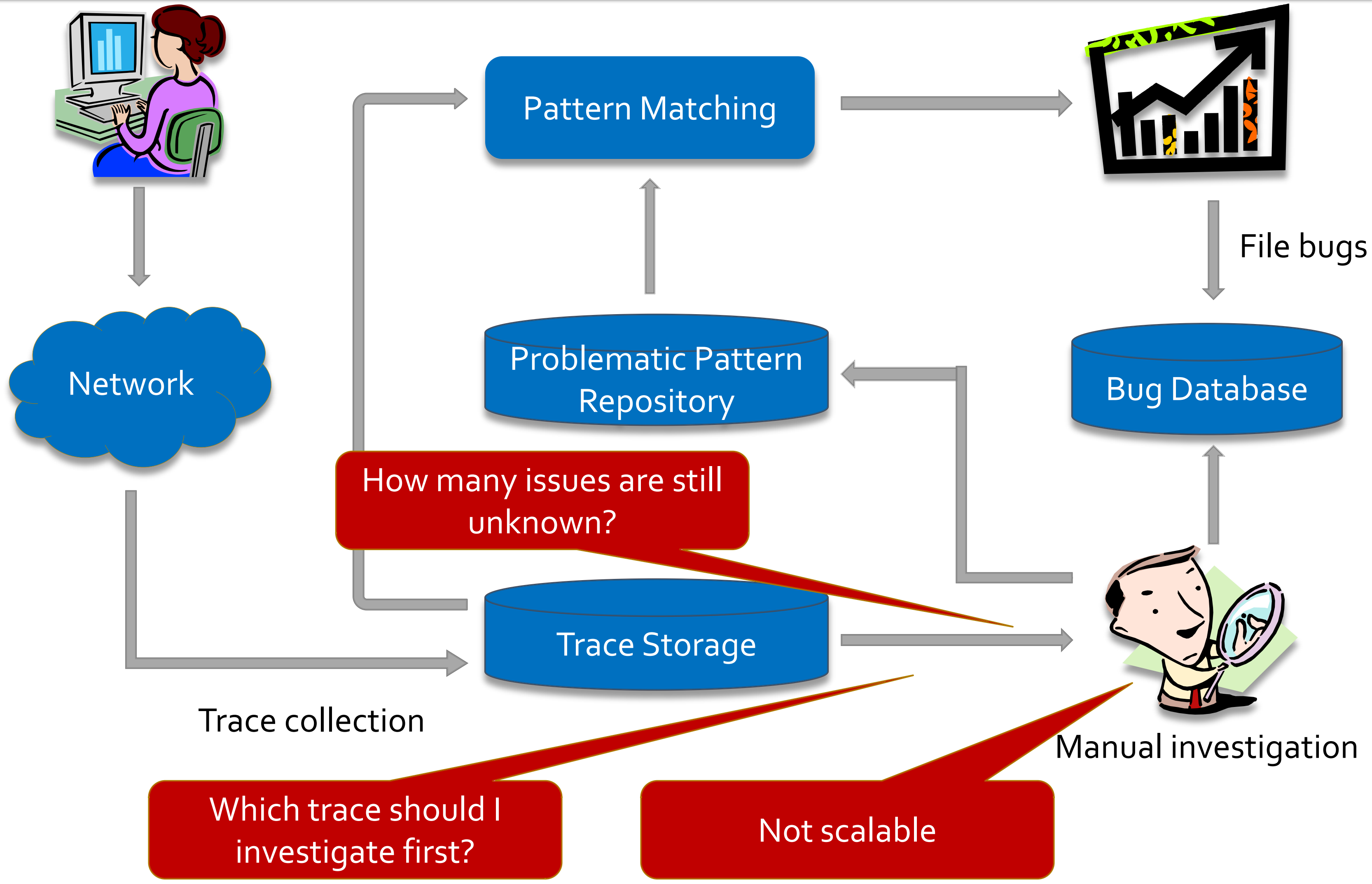
- Large scale trace data
- Highly complex performance analysis in OS level
- Combination of machine learning and domain expertise



Formulation of Callstack Mining/Clustering Problem

- Unknown issue discovery
- Issue prioritization
- Scalable to large number of traces

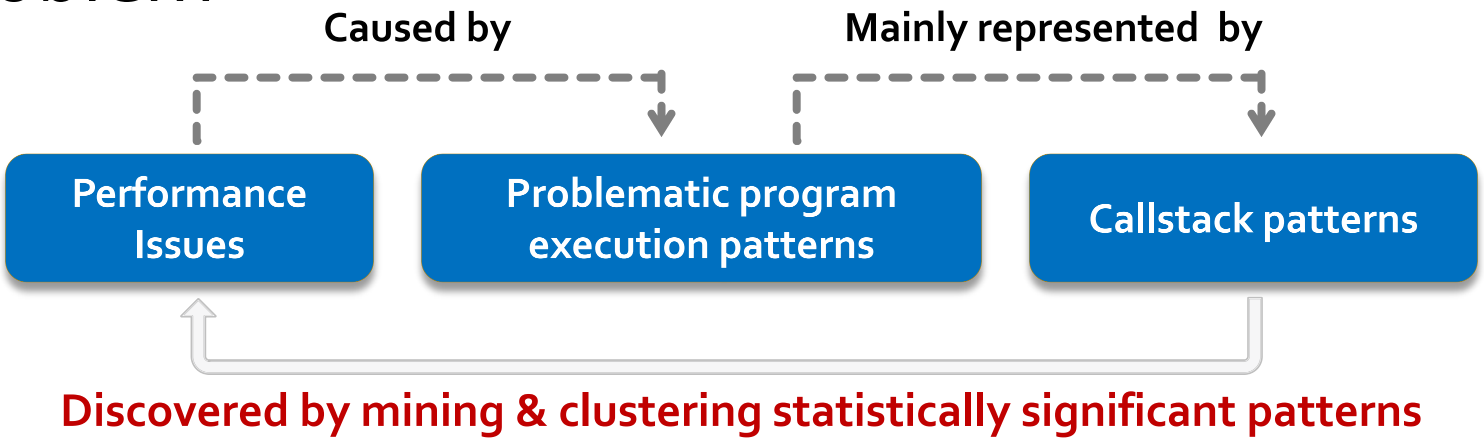
Performance Related Trace Analysis Today



StackMine Approach

[ICSE 12] in collaboration with MSR Asia

- Formulate as a callstack mining and clustering problem


Caused by
Mainly represented by
Performance Issues
Problematic program execution patterns
Callstack patterns
Discovered by mining & clustering statistically significant patterns

- Incorporate deep domain knowledge

Industry Impact

"We believe that the MSRA tool is highly valuable and much more efficient for mass trace (100+ traces) analysis. For 1000 traces, we believe the tool saves us 4-6 weeks of time to create new signatures, which is quite a significant productivity boost."

- Development Manager in Windows

Effective discovery of new issue on Windows mini-hang

Continuous impact on future Windows versions

Tool-Tool Cooperation

- Static analysis + dynamic analysis
  - Static Checker + Test Generation
  - …
- Dynamic analysis + static analysis
  - Fix generation + fix validation
  - …
- Static analysis + static analysis
  - …
- Dynamic analysis + dynamic analysis [ASE 08]
  - …

Conclusion: Cooperative Testing and Analysis

- Human-Assisted Computing
  - Tool → Human: expose more/less details?
  - Tool ←Human: not reliable guidance?
- Human-Centric Computing
  - Human → Tool: more input modalities?
  - Human ← Tool: tutoring hints?
- Human-Human
- Computing-Computing

Thank you!

Questions ?

https://sites.google.com/site/asergrp/

Automated Software Engineering Research Group @NCSU