

Out of the Ivory Tower: Are We There Yet on Automatic Test Data Generation?

Tao Xie

North Carolina State University
Raleigh, NC, USA

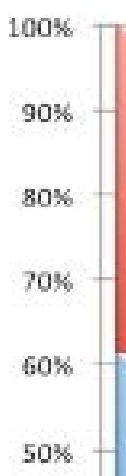
In Collaboration with Microsoft Research Redmond/Asia, PKU, Students@NCSU ASE Group

ICSE Papers: Industry vs. Academia

OSDI 2008 26% vs. xSE ?%

Developers, Programmers, Architects Among All Attendees

average



ICSE 09 Keynote



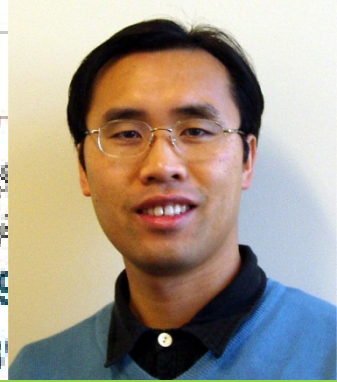
ICSM 11 Keynote



MSR 11 Keynote



MSR 12 Keynote



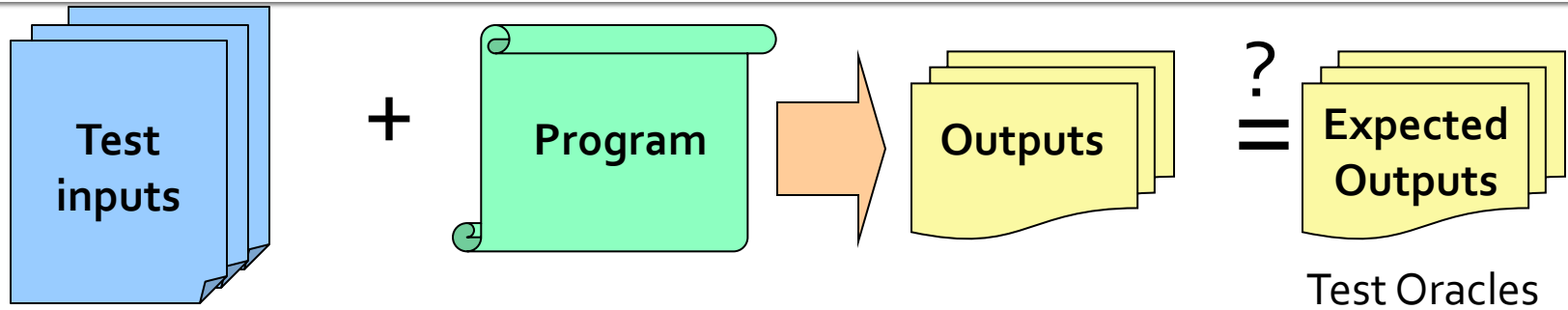
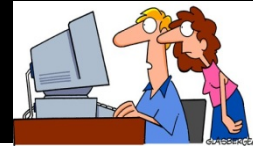
SCAM 12 Keynote

average 1976-1994
AC 56%
IND 44%

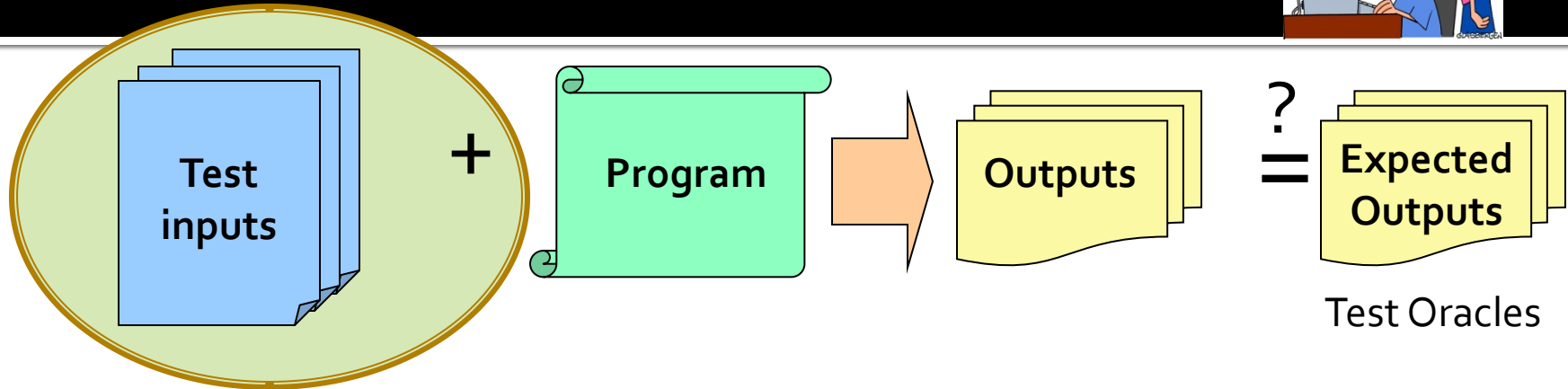
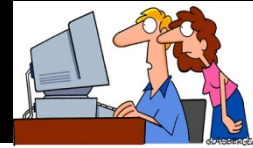
average 1995-2008
AC 83%
IND 17%

Source© Carlo Ghezzi

Software Testing Setup

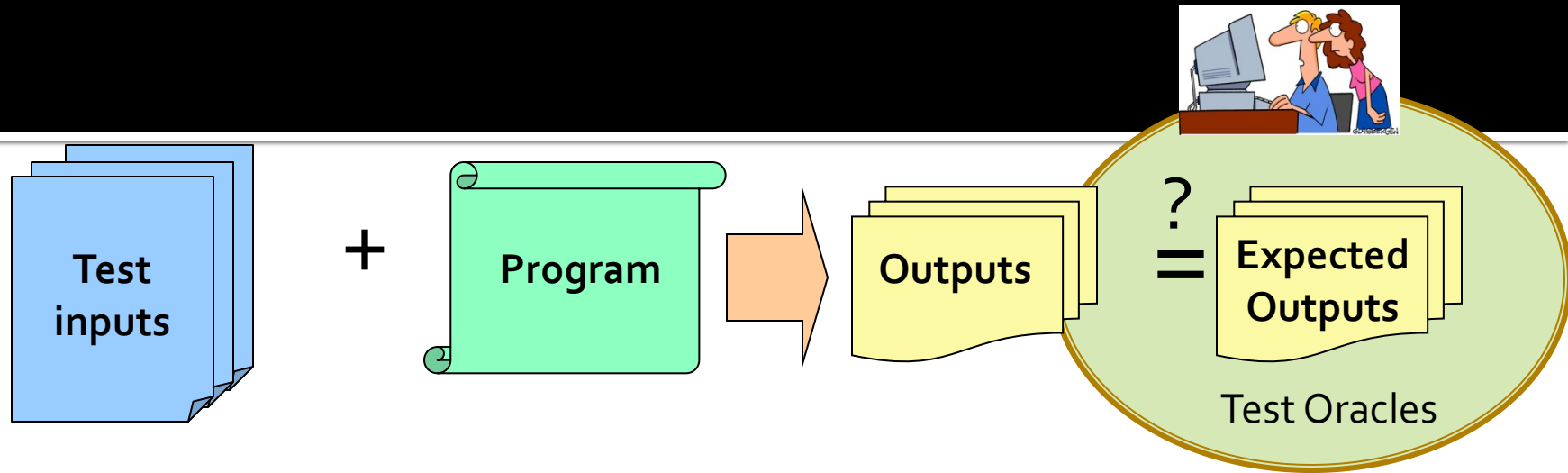


Software Testing Problems



- **Test Generation:** Generate test inputs of high quality (e.g., high structural coverage, high fault-detection capability)

Software Testing Problems



- **Test Generation:** Generate test inputs of high quality (e.g., high structural coverage, high fault-detection capability)

- **Test-Oracle Construction:** Construct test oracles of high quality (e.g., high fault-detection capability)

Research on

Automatic Test Data Generation (ATDG)

- Many years of research
- Many testing researchers
- Many published papers
 - E.g., 14 papers out of 31 papers at ISSTA 2012
- Many released tools

- How well has research been transferred to industrial practice?

Tech-Adoption Evidence in Literature

- Papers on industrial studies/evaluations on applying tools on industrial code, who apply?
 - **Authors themselves instead of third parties**
 - Non-target users (such as students)
 - Target users but not developers of the industrial code
 - **Developers of the industrial code**
- Apply one-time (hit&run) or **continuous adoption**?
 - Good example on continuous industrial adoption:
MSRA Software Analytics: StackMine [ICSE 12], XIAO [ACASC 12], ...

<http://research.microsoft.com/en-us/groups/sa/>

Agitar AgitarOne – Developer Testing

The screenshot shows the Agitar IDE interface. The Package Explorer on the left displays the project structure, including the 'src' directory and various Java files. The Source Editor in the center shows the 'Product.java' file with the following code:

```
/**
 * @param code Must be of the form A-9999-99-A
 * @param name Must not be null. Must not be longer than 20 characters.
 * @param price must be less than $1000
 * @throws IllegalArgumentException if the code, name or price is invalid
 */
public Product(String code, String name, double price) throws IllegalArgumentException {
    validateCode(code);
    validateName(name);
    validatePrice(price);
    this.code = code;
    this.name = name;
    this.price = price;
}

public String getCode() {
    return code;
}
```

The Console/Agitation window at the bottom shows the test results for the 'Product' class. The test suite is 'Product -> <init>(String, String, double) -> NORMAL'. The test results are as follows:

Value	True	False
0.0 <= this.getPrice() <= 1000.0	42	0
this.getCode().length() == 11	42	0
this.getPrice() == price	42	0
this.getName() == name	42	0
this.getCode() == code	42	0
1 <= this.getName().length() <= 20	42	0

Image from <http://www.agitar.com/>

Boshernitsan, Doong, and Savoia. From Daikon to Agitator: lessons and challenges in building a commercial tool for developer testing. ISSTA 2006.

Microsoft Research : SpecExplorer

Protocol Doc Interoperability Testing

- 222 protocols/technical documents tested
- 22,847 pages studied and converted into requirements
- 36,875 testable requirements identified and converted into test assertions
 - 69% tested using Model-Based Testing
 - 31% tested using traditional test automation
- 66,962 person days (250+ years)
 - Hyderabad: 250 test engineers
 - Beijing: 100 test engineers

From talk slides by Yiming Cao

Microsoft Research

SAGE – Security Fuzz Testing

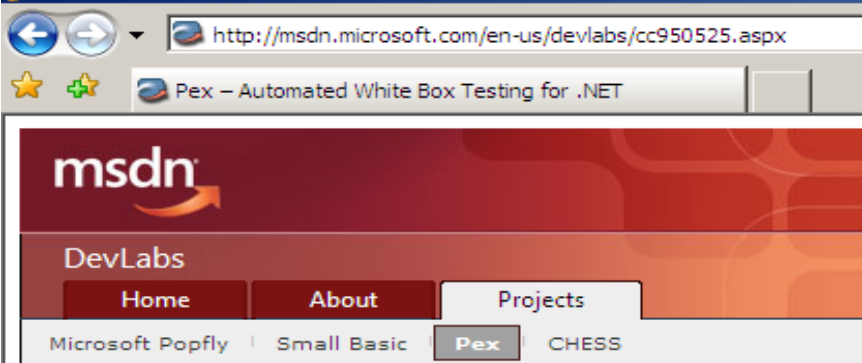
- Since April 2007 1st release: many new security bugs found (missed by blackbox fuzzers, static analysis)
- Example: Windows Client Security team for Win7
 - Dedicated fuzzing lab with 100s machines ☐
 - 100s apps (deployed on 1 billion+ computers)
 - ~1/3 of all fuzzing bugs found by SAGE

Microsoft Research

Pex/Fakes - Developer Testing



Pex - Automated White Box Testing for .NET - Windows Internet Explorer

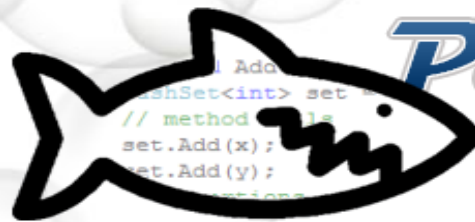


Pex download counts (20 months)
(Feb. 2008 - Oct. 2009)

Academic: **17,366**

Devlabs: **13,022**

Total: 30,388



Pex

Automated White Box Testing for .NET

	value
1	null
2	""
3	\0

Fakes



Visual Studio®

Premium & Ultimate 2012 Overview

About Pex - Automated White Box Testing for .NET

Pex (Program EXploration) produces a traditional unit test suite. It takes a code snippet, its parameters, calls the code under test, and states assertions. Pex automatically generates 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. This is useful for testing an extremely well-tested component.

<http://research.microsoft.com/projects/pex/>


Microsoft Research *Pex for Fun*

Teaching/Learning CS via Interactive Gaming

www.pexforfun.com

Curious? Learn More! →

Pex



Coding Duel
for fun

My Duels ▾ | Settings ▾ | Sign In

Random Puzzle Learn New

1,019,768 clicked 'Ask Pex!'

C# Visual Basic F#

This puzzle is an interactive Coding already won this Duel 305 times! [Help](#)

Other people have

using System;

```
public class Program {  
    public static int Puzzle(int x) {  
        // Can you write code to solve the puzzle?  
        return x;  
    }  
}
```

Over **1 million** game-play interactions made by players around the world since Summer 2010

Ask Pex!

Non-Success Factors?

- Unavailability of tools in the public domain
 - TOSEM papers 2001-2006 (60% refer to a tool, only 20% **installable**) [Ghezzi ICSE 09 keynote]
- Evaluation subjects (scalability)
 - Toy examples (prior 2000) → hand-picked isolated complex data structures [Korat ISSTA 02] → open source real-world code [MSeqGen/Seeker FSE 09/OOPSLA 11]
- Self referentiality
 - E.g., compare proposed X-type test-gen technique within only other X-type test-gen techniques w/o comparing with other test-gen techniques (X: random, evolutionary, DSE, ...)
- Usability ignored (assuming human not in the loop)
- Company/practitioner culture in practice

Summary

- Status of SE research community (e.g., ICSE)
- Zoom in: seeking evidence of tech adoption of ATDG in practice
- Some success stories with varied levels of extent
- Some non-success factors

- What to do next to deal with the gap issues? ...
 - Panel discussion next session

Thank you!

Questions ?



<https://sites.google.com/site/asergroup/>

Automated
Software Research
Group
Engineering@NCSU

Behind the Scene of Pex for Fun



Secret Implementation

```
class Secret {  
    public static int Puzzle(int x) {  
        if (x <= 0) return 1;  
        return x * Puzzle(x-1);  
    }  
}
```

```
class Test {  
    public static void  
        if (Secret.Puzzl  
            throw new E  
    }  
}
```

Secret Impl ^{behavior} == Player Impl

Player Implementation

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

Ask Pex!



	x	your result	secret implementation result	Output/Exception
✓	1	1	1	
✓	2	2	2	
✗	3	3	6	Mismatch

What Make Tech Transfer Difficult?

- Scalability
- Complexity
- Applicability
- Usability (human in the loop)
- Cost-Benefit Analysis

Scalability

- Academia
 - Rarely ask “When scale is up, will my solution still work?”
 - Tend to focus on small or toy scale problems
- Real-world (e.g., search engine, code analysis, ...)
 - Often demand a scalable solution
- Ideal: sophisticated and scalable solution
 - But in practice, simple solution tends to be scalable (performance, maintenance, ...)
 - Academia tend to value sophistication > simplicity
- Ex: Test prioritization@Microsoft [ISSTA 2002], Klee [OSDI 2008]

Complexity

■ Academia

- Tend to make assumptions to simplify problems, or one at a time (indeed relaxing assumptions over time)
- May not be able to assess the relevance/feasibility of assumptions in practice; not consult/work w/ industry

■ Real-world

- Often has high complexity, violating these assumptions

■ Example: OO Unit Test Generation

- Isolated simple classes → Isolated complex data structures → Real world classes as focused by our recent work [ESEC/FSE 2009, OOPSLA 2011]

Applicability

■ Academia

- Tend to focus on a solution optimized for one of many situations (likely worse for others) vs. comprehensive solution
- May not enable to tell ahead of time whether a given case would fall into applicable scope of the solution

■ Real-world

- Need a comprehensive solution that would work generally (at least not compromising too much other situations)

■ Examples

- Integration of our Fitnexus in Pex [DSN 2009]
- Coverity [CACM 2010] vs. MSRA XIAO/PatternInsight
- Industry adoption of open source tools

Usability

■ Academia

- Tend to leave human out of loop (involving human makes evaluations difficult to conduct or write)
- Tend not to spend effort on improving tool usability
 - tool usability would be valued more in HCI than in SE
 - too much to include both the approach/tool itself and usability/its evaluation in a single paper

■ Real-world

- Often has human in the loop (familiar IDE integration, social effect, lack of expertise/willingness to write specs,...)

■ Examples

- Agitar [ISSTA 2006] vs. Daikon [TSE 2001]
- Debugging user study [ISSTA 2011]

Cost-Benefit Analysis

■ Academia

- Tend to focus on one or a few dimensions of measurement (e.g., analysis cost, precision and/or recall)

■ Real-world

- Consider many dimensions of measurement
 - Cost, e.g., human cost
 - Benefit, e.g., bug severity

■ Example

- FindBugs experience at Google [ISSTA 2009]

Suggestions

- Value engineering creativity
- Find killer apps, e.g.,
 - MSR SLAM: Device driver verification
 - MSR Sage: Security testing of binaries
 - PatternInsight/MSRA Xiao: Known-bug detection
- Engage practitioners
 - Get research problems from real practice
 - Get feedback from real practice
 - Collaborate across disciplines
 - Collaborate with industry

Industry Academia Collaboration

- Academia (research recognitions, e.g., papers) vs. Industry (company revenues)
- Academia (research innovations) vs. Industry (likely involving engineering efforts)
- Academia (long-term/fundamental research) vs. Industry (short-term research or work)
- ...
- Industry: problems, infrastructures, data, evaluation testbeds, ...
- Academia: educating students, ...