Evolving Testing and Analysis for Evolving Software
From Ivory Tower to Real World

Tao Xie

Peking University (2011-2012), China
North Carolina State University
Raleigh, NC, USA

In Collaboration with Microsoft Research Redmond/Asia, and Students@NCSU ASE Group
Automation in Software Engineering

ASE 2010
25th IEEE/ACM International Conference on Automated Software Engineering

ASE 2011
26th IEEE/ACM International Conference on Automated Software Engineering

Automated Software Engineering Research Group @ NCSU
Automation in Software Testing

ACM SIGSOFT International Symposium on Software Testing and Analysis

Dagstuhl Seminar 10111

Practical Software Testing: Tool Automation and Human Factors
ICSE Papers: Industry vs. Academia

OSDI 2008 26% vs. xSE ?%
Developers, Programmers, Architects Among All Attendees

ICSE 2009 Keynote

ICSM 2011 Keynote

Source© Carlo Ghezzi
50 years of automated debugging research
- N papers → only 5 evaluated with actual programmers

"Programmers have been waiting a long time for usable automated debugging tools, and we have already gone a long way from the early days of debugging. We believe that, to further advance the state of the art in this area, we must steer research towards more promising directions that take into account the way programmers actually debug in real scenarios."

[ISSTA11 Parnin&Orso]
Are Regression Testing [Research] Techniques Actually Helping Industry?

- Likely most studied testing problems
  - N papers

“However, empirical evaluation and application of regression testing techniques at industrial level seems to remain limited [182]. Out of the 159 papers listed in Table IV, V, VI and VII, only 31 papers list a member of industry as an author or a co-author. More importantly, only 12 papers consider industrial software artefacts as a subject of the associated empirical studies [68, 69, 88, 99–102, 104–106, 126, 147]. This suggests that a large scale industrial uptake of these techniques has yet to occur.”
From Ivory Tower to Real World

**Unsuccessful** cases/experiences on tech transfer/adoption

**Successful** cases/experiences on tech transfer/adoption

Learning by Negative Examples

Learning by Positive Examples

Using Industrial Artifacts ≠ Technology Adoption
Outline

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]
- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]
- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]
- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
Outline

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]
- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]
- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]
- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
public class BST implements Set {
    Node root;
    int size;
    static class Node {
        int value;
        Node left;
        Node right;
    }
    public void insert (int value) { ... }
    public void remove (int value) { ... }
    public bool contains (int value) { ... }
    public int size () { ... }
}
Example Generated Tests

Test 1 (T1):
BST t1 =
    new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

Test 2 (T2):
BST t2 =
    new BST();
t2.insert(2);
t2.insert(3);

Test 3 (T3):
BST t3 =
    new BST();
t3.insert(2);
t3.insert(1);
t3.size();

Each test has a method sequence on the objects of the class.
Defining Redundant Tests

- Test $T$ is redundant w.r.t. the existing tests if $T$ exercises no new program behavior (thus, no new bug)
- Difficulties with redundant tests
  - costly to generate, run, and inspect them
  - may provide false confidence
- Rostra characterizes behavior using input values
  - identify and remove redundant tests
Previous Approaches

- Behavior characterized using code coverage (e.g., statements, branches)
  - False positives: remove many non-redundant tests
  - Reduced test suite can be dramatically worse w.r.t. fault detection capability [Rothermel et al. 98, Jones&Harrold 03]

- Behavior characterized using method sequence
  - False negatives: fail to remove many redundant tests
  - Adopted by most existing test generation tools

Test 1:
BST t1 = new BST();
t1.size();

Test 2:
BST t2 = new BST();
t2.size();
t2.size();
Example Generated Tests

Test 1 (T1):
BST t1 =
    new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

Test 2 (T2):
BST t2 =
    new BST();
t2.insert(2);
t2.insert(3);

Test 3 (T3):
BST t3 =
    new BST();
t3.insert(2);
t3.insert(1);
t3.size();

Approach using method sequence removes no tests

Rostra removes T2 and T3 because T2 and T3 are redundant w.r.t. T1
Focus on each method execution individually

Unnecessary to test a method with the same inputs (same inputs $\Rightarrow$ same behavior)
  - deterministic method execution: no randomness, no multithreading interaction
  - method inputs: incoming program states
    - receiver-object state: transitively-reachable-field values
    - arguments
    - accessed static fields
Running a test produces a set of method executions

A test is **redundant** w.r.t. a test suite

- if the method executions produced by the test is a subset of the method executions produced by the test suite
Detecting Redundant Tests

Test 1 (T1):
BST t1 =
    new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

Test 2 (T2):
BST t2 =
    new BST();
t2.insert(2);
t2.insert(3);

Method Exec

receiver-obj state

argument

new BST()
Detecting Redundant Tests

Test 1 (T1):
BST t1 =
   new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

Test 2 (T2):
BST t2 =
   new BST();
t2.insert(2);
t2.insert(3);

Method Exec

<table>
<thead>
<tr>
<th>receiver-obj state</th>
<th>argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>root = null</td>
<td>2</td>
</tr>
<tr>
<td>size = 0</td>
<td></td>
</tr>
</tbody>
</table>

The method `BST.insert` is exercised.
**Test 1 (T1):**

BST t1 =
new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

**Test 2 (T2):**

BST t2 =
new BST();
t2.insert(2);
t2.insert(3);
t2.insert(3);

**Method Exec**

<table>
<thead>
<tr>
<th>receiver-obj state</th>
<th>argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>root = 2</td>
<td>3</td>
</tr>
<tr>
<td>size = 1</td>
<td></td>
</tr>
</tbody>
</table>

BST.insert
Detecting Redundant Tests

Test 1 (T1):
BST t1 =
    new BST();
t1.insert(2);
t1.insert(1);
t1.remove(1);
t1.insert(3);
t1.size();

Test 2 (T2):
BST t2 =
    new BST();
t2.insert(2);
t2.insert(3);

Test 2 is redundant w.r.t Test 1!
Evaluation Results [ASE '04]

- Industry standard tool adopting previous approach based on method sequences
  - Parasoft Jtest 4.5 www.parasoft.com
    - Generate tests with method-call lengths up to three
- Use Jtest to generate tests for 11 Java classes from various sources (complex data structures)
- Apply Rostra on the generated tests

- 90% of generated tests are redundant!
- Minimized tests preserve the same code (branch) coverage and seeded- bug coverage
Industry Impact — Parasoft Jtest

- People do use Jtest
  - Recognized with numerous awards, including Jolt Product Excellence Award and JDJ Editor's Choice Award in 2004; adopted by thousands of development teams worldwide.
    
    — businesswire.com

- But don’t love its test generation
  - “I can't think of anyone telling me that they love Jtest's test-generating feature.”
    
    —Joe Rainsberger, JUnit book author, 02/05@junit user mailing list
Industry Impact — Parasoft Jtest

- People do use Jtest
  - Recognized with numerous awards, including Jolt Product Excellence Award and JDJ Editor's Choice Award in 2004; adopted by thousands of development teams worldwide.
    — businesswire.com

- And do love its test generation
  - “I can't think of anyone telling me that they love Jtest's test-generating feature.”
    — Joe Rainsberger, JUnit book author, 02/05@junit user mailing list

Parasoft VP later notified us that Parasoft Jtest 6.0 had fixed the test redundancy issue identified by us
Outline

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]
- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]
- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]
- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
Outline

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]

- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]

- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]

- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
void CoverMe(int[] a)
{
    if (a == null) return;
    if (a.Length > 0)
        if (a[0] == 1234567890)
            throw new Exception("bug");
}
Challenges of DSE

- **Loops**
  - Fitnex [DSN 09]

- **Method sequences**
  - MSeqGen [ESEC/FSE 09], Seeker [OOPSLA 11]

- **Environments**
  - Database [ASE 09-sp, ASE 11], Cloud [IEEE Soft 12]

**Opportunities**

- Regression testing [ISSTA 11]
- Developer guidance (cooperative testing) [ICSE 11]
**Pex on MSDN DevLabs**

**Incubation Project for Visual Studio**

Download counts (20 months)

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

---

About Pex – Automated White Box Testing for .NET

Pex (Program EXploration) produces a traditional unit test suite with high code coverage. A parameterized unit test is simply a method that takes 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.
Open Source Pex extensions
http://pexase.codeplex.com/


Pex Extensions: Automated Software Engineering Group@NCSU

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
Outline

- **Play Around Industrial Tool**
  - Parasoft Jtest ➔ Rostra [ASE 04]

- **Play Within Industrial Tool**
  - Microsoft Research Pex ➔ Fitnex [DSN 09]

- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]

- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
When are Tools Worse Than Human?
"Completely Automated Public Turing test to tell Computers and Humans Apart"
Automation in Software Testing

ACM SIGSOFT International Symposium on Software Testing and Analysis

Human Factors

Dagstuhl Seminar 10111

Practical Software Testing: Tool Automation and Human Factors
Dagstuhl Seminar 10111
Practical Software Testing: Tool Automation and Human Factors

Human Factors
Human-Centric Computing in Software Engineering

The IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)

ACM Symposium on Software Visualization

http://www.softvis.org
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 ∪ B?
Cooperation Between Human and Machine: Cooperative Testing/Analysis

- **Computing-Centric Human**
  - Driver: tool ↔ Helper: human
  - Ex. Covana [Xiao et al. ICSE 2011]

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

*Interfaces are important. Contents are important too!*
Computing-Centric Human

- **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-Test-Generation Tool

external-method call problems (EMCP)

object-creation problems (OCP)
Cooperation Between Human and Machine – Covana

- **Task:** What need to automate?
  - Test-input generation

- **What difficulties does the tool face?**
  - Doesn’t know which methods to instrument and explore
  - Doesn’t know how to generate effective method sequences

- **How to communicate info to the user to get her help?**
  - Report encountered problems

- **How does the user help the tool based on the info?**
  - Instruct which external methods to instrument/write mock objects
  - Write factory methods for generating objects

- **Iterations to form feedback loop?**
  - Yes, till the user is happy with coverage or impatient

[Xiao et al. ICSE 2011]
Computing-Centric Human
- Driver: computer ↔ Helper: human
- Ex. Covana [Xiao et al. ICSE 2011]

Human-Centric Computing
- Driver: human ↔ Helper: computer
- Ex. Coding duels @Pex for Fun

Interfaces are important. Contents are important too!
Behind the Scene of Pex for Fun

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;
    }
}

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.
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!

```csharp
using System;

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;
    }
}
```
Coding duels at http://www.pexforfun.com/

**Task** for Human: write behavior-equiv code

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

**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-Centric Computing

- Coding duels
  - Brain exercising
  - Fun: iterative, adaptive/personalized, w/ win criterion
  - Abstraction/generalization, debugging, problem solving

Coding duels at http://www.pexforfun.com/

Fun: iterative, adaptive/personalized, w/ win criterion

Abstraction/generalization, debugging, problem solving
Coding Duels Go Wild @ICSE 11

Data-Driven Software Engineering in the Large AND Small
Outline

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]

- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]

- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]

- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
Outline

- **Play Around Industrial Tool**
  - Parasoft Jtest ➔ Rostra [ASE 04]
- **Play Within Industrial Tool**
  - Microsoft Research Pex ➔ Fitnex [DSN 09]
- **Advise Industrial Tool Developers**
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]
- **Engage Target Users**
  - Microsoft Research Asia Software Analytics Group
Utilize data-driven approach to help create highly performing, user friendly, and efficiently developed and operated software and services.

Software Analytics as a Learning Case in Practice: Approaches and Experiences

Dongmei Zhang¹, Yinguong Dang¹, Jian-Guang Lou¹, Shi Han¹, Haidong Zhang¹, Tao Xie²

¹Microsoft Research Asia, Beijing, China
²North Carolina State University, Raleigh, NC, USA
{dongmeiz,yidang,jlou,shihan,haizhang}@microsoft.com, xie@csc.ncsu.edu

ABSTRACT

Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services. In this position paper, we advocate that when applying analytic technologies in practice of software analytics, one should (1) incorporate a broad spectrum of domain knowledge and expertise, analysis in order to obtain insightful and actionable information for data-driven tasks around software and services.

Insightful information is information that conveys meaningful and useful understanding or knowledge towards performing the target task. Typically insightful information is not easily attainable by directly investigating the raw data without aid of analytic technologies. Actionable information is information upon which software

http://research.microsoft.com/groups/sa/

[MALETS’11 Zhang et al.]
Motivation
- Copy-and-paste is a common developer behavior
- A real tool widely adopted at Microsoft

XIAO enables code clone analysis with
- High tunability
- High scalability
- High compatibility
- High explorability
Successful Tech-Transfer Tips

- Engagement of practitioners
- Combination of expertise

What Shall Academia Do?

- Get research problems from real practice
- Get feedback from real practice
- Collaborate across disciplines
- Collaborate with industry
Summary

- Play Around Industrial Tool
  - Parasoft Jtest ➔ Rostra [ASE 04]
- Play Within Industrial Tool
  - Microsoft Research Pex ➔ Fitnex [DSN 09]
- Advise Industrial Tool Developers
  - Microsoft Research Pex For Fun ➔ [CSEE&T 11 Tut]
- Engage Target Users
  - Microsoft Research Asia Software Analytics Group
Vision: Cooperative Testing/Analysis

- **Computing-Centric Human**: Test/Analysis Tools
  - Tool → Human
  - Tool ← Human

- **Human-Centric Computing**: Educational Tools
  - Human → Tool
  - Human ← Tool

- **Computing-Computing** (synergetic analysis)

- **Human-Human** (crowdsourcing)
Thank you!

Questions?

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