Welcome to today’s ACM SIGSOFT Webinar
Transferring Software Testing Tools to Practice

• The presentation starts at the top of the hour and lasts 60 minutes. Slides will advance automatically throughout the event. You can resize the slide area as well as other windows by dragging the bottom right corner of the slide window, as well as move them around the screen.

• If you are experiencing any problems/issues, refresh your console by pressing the F5 key on your keyboard in Windows, Command+R if on a Mac, or refresh your browser if you’re on a mobile device; or close and re-launch the presentation. You can also view the Webcast Help Guide, by clicking on the “Help” widget in the bottom dock.

• If you think of a question during the presentation, please type it into the Q&A box and click on the submit button. You do not need to wait until the end of the presentation to begin submitting questions.

• You can download a copy of these slides by clicking on the Resources widget in the bottom dock.

• This session is being recorded and will be archived for on-demand viewing in the next 1-2 days. You will receive an automatic email notification when it is available.

• Information on future and past webinars is found at:
  • http://www.sigsoft.org/resources/webinars.html
  • http://learning.acm.org/webinar
Transferring Software Testing Tools to Practice

Nikolai Tillmann
Microsoft

Tao Xie
University of Illinois, Urbana-Champaign

Pratap Lakshman
Microsoft

Judith Bishop
Microsoft Research
(Automated) Test Generation

```java
public static TriangleKind ClassifyBySideLengths(int[] lengths) {
    //validate(lengths);
    ...
}
```

- Human
  - Expensive, incomplete, ...
- Brute Force
  - Pairwise, predefined data, etc...

- Tool Automation!!
Dynamic Symbolic Execution

[DART: Godefroid et al. PLDI’05]

Code to generate inputs for:

```csharp
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.Length>0) |
a[0]!=1234567890 | a!=null && a.Length>0 && a[0]!=1234567890 |
{123} | a!=null && a.Length>0 && a[0]==1234567890 |

Done: There is no path left.
Microsoft Research Automated Test Generation Tool: Pex & Relatives

• **Pex** (released in May 2008)

http://research.microsoft.com/pex/
Microsoft Research Automated Test Generation Tool: Pex & Relatives

• **Pex** (released in May 2008)
  • 30K downloads after 20 months
  • Active user community: 1.4K forum posts during 3 years
  • Shipped with Visual Studio 2015 as IntelliTest

• **Moles** (released in Sept 2009)
  • Shipped with Visual Studio 2012 as Fakes
  • “Provide Microsoft Fakes w/ all Visual Studio editions” got 1.5K community votes

What went on behind the scenes to build a user base?

http://research.microsoft.com/pex/
Lesson 1. Evolving Vision

Parameterized Unit Tests Supported by Pex

```csharp
void TestAdd(ArrayList a, object o) {
    Assume.IsTrue(a!=null);
    int i = a.Count;
    a.Add(o);
    Assert.IsTrue(a[i] == o);
}
```

- **Surrounding** (Moles/Fakes)
- **Retargeting** (Pex4Fun/Code Hunt)
- **Simplifying** (IntelliTest)
Lesson 2. Landing the First Customer

Macro Perspective

Tool Adoption by (Mass) Target Users

Tool Shipping with Visual Studio

Micro Perspective

- Developer/manager: “Who took a dependency on your tool?”
- Pex team: “Do you want to be the first?”
- Developer/manager: “I love your tool but no.”
Lesson 2. Landing the First Customer

• Tackle real-world challenges
  • Demo Pex on real-world cases (e.g., ResourceReader) beyond textbook examples
  • Demo Moles to well address important scenarios (e.g., unit testing SharePoint code)

• Address technical/non-technical barriers for tech adoption in industry
  • Offer tool license not prohibiting commercial use

• Incremental shipping
  • Ship experimental reduced versions and gather feedback

• Find early adopters

• Provide quantitative info (reflecting tool’s importance or benefit extent)
  • Not all downloads are equal! (e.g., those from Fortune 500)
Lesson 3. Human Factors – Generated Data Consumed by Human

• **Developer**: “Code digger generates a lot of “\0” strings as input. I can’t find a way to create such a string via my own C# code. Could any one show me a C# snippet? I meant zero terminated string.”

• **Pex team**: “In C#, a \0 in a string does not mean zero-termination. It’s just yet another character in the string (a very simple character where all bits are zero), and you can create as Pex shows the value: “\0”.”

• **Developer**: “Your tool generated “\0””

• **Pex team**: “What did you expect?”

• **Developer**: “Marc.”
Lesson 3. Human Factors – Generated Name Consumed by Human

- **Developer**: “Your tool generated a test called Capitalize01. I don’t like it.”
- **Pex team**: “What did you expect?”
- **Developer**: “Capitalize_Should_Fail_When_Value_Is_Null.”
Lesson 3. Human Factors – Generated Results Consumed by Human

Object Creation messages suppressed (e.g., Covana by Xiao et al. [ICSE’11])

Exception Tree View

Exploration Tree View

Exploration Results View
Lesson 4. Misconceptions

• Someone advertises: “Simply one mouse click and then everything would work just perfectly”
  • Often need environment isolation w/ Moles/Fakes

• “One mouse click, a test generation tool would detect all or most kinds of faults in the code under test”
  • Developer: “Your tool only finds null references.”
  • Pex team: “Did you write any assertions?”
  • Developer: “Assertion???”

• “I do not need test generation; I already practice unit testing (and/or TDD). Test generation does not fit into the TDD process”
Lesson 5. Embracing Feedback

Gathered feedback from **target tool users:**

- **Directly,** e.g., via
  - MSDN Pex forum, tech support, outreach to MS engineers and .NET user groups, outreach to external early adopters

- **Indirectly,** e.g., via
  - interactions with the Visual Studio team (a tool vendor to its huge user base)

- Lack of basic test isolation in practice => **Moles**
  - Our suggestion of refactoring code for testability faced strong resistance in practice
  - Observation at Agile 2008 conference
    - Large focus on mock objects and tool support for mock objects
Pex to IntelliTest - Product Group Point of View

*Productizing*

- Articulate product alignment and value proposition
- (Re)Establish value with the broader community
  - Gather feedback and experiences from early practitioners
  - Dev Labs extensions on Visual Studio 2013
- Ship layered experiences
  - Explore, Persist, Assist, Specify
- Refine the user interface
  - Conform with Visual Studio look-n-feel
  - Explain/reconcile how the generated code looks like
Gathering Feedback

Feedback

- Early drops on VS Code Gallery of the Pex Extension, and the Code Digger extensions for Visual Studio 2013
- Visual Studio MVP Community
- Internal dogfooding by teams within Microsoft
- Uservoice feedback (> 20 ideas)

Twitter
https://twitter.com/pexandmoles

Stackoverflow
Active forum with questions tagged with "Pex" or "IntelliTest"

Facebook
https://www.facebook.com/PexMoles/

Item | Votes
--- | ---
Add support for NUnit and xUnit.net | 304
Add Support for VB.NET | 336
Make it available with Visual Studio Professional | 319
Enable IntelliTest for 64 bit projects | 72
Pex to IntelliJTest - Adds and Cuts

• Additions
  • Externalizing Test Framework support

• Cuts
  • Visual Basic Support
  • CommandLine Support
  • FixIt
  • Code Contract integration
  • Pex Explorer
  • Reporting
Pex to IntelliTest - *Shipped!*

NUnit Extension – 13024 downloads

xUnit.net Extension - 8589 downloads


**Why is it as stable as we claim?**

We have used Visual Studio IntelliTest (Announcement for Visual Studio 2015: [https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx](https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx)) to extensively test some important properties of this BBCode-Parser. We used IntelliTest to ensure that the parser never crashes and that it never emits any dangerous tag such as...
Collaboration with Academia

• **Win-win** collaboration model
  • Win (Industry Lab): longer-term research innovation, man power, research impacts, ...
  • Win (University): powerful infrastructure, relevant/important problems in practice, both research and industry impacts, ...

• Hosting academic visitors
  • Faculty visits
e.g., Fitnex [Xie et al. DSN’09], Pex4Fun [Tillmann et al. ICSE’13 SEE]
  • Student internships
e.g., FloPSy [Lakhotia et al. ICTSS’10], DyGen [Thummalapenta et al. TAP’10]

http://research.microsoft.com/pex/community.aspx#publications
Engaging Broader Academic Communities

• Academic research **inspiring** internal technology development
  • Reggae [Li et al. ASE’09] → Rex [Veanes et al. ICST’10]
  • MSeqGen [Thummalapenta et al. FSE’09] → DyGen [Thummalapenta et al. TAP’10]
  • ...

• Academic research **exploring** longer-term research frontiers
  • DySy [Csallner et al. ICSE’08]
  • Seeker [Thummalapenta et al., OOPSLA’11]
  • Covana [Xiao et al. ICSE’11]
  • SEViz [Honfi et al. ICST’15]
  • Pex + Code Contracts [Christakis et al. ICSE’16]
  • ...

http://research.microsoft.com/pex/community.aspx#publications
Going from Pex to Coding Duels

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

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("Mismatch");
    }
}
```

<table>
<thead>
<tr>
<th>x</th>
<th>your result</th>
<th>secret implementation result</th>
<th>Output/Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>
About Code Hunt

Powerful and versatile platform for coding as a game
Built on the symbolic execution of Pex
Addressing different audiences – students, developers, researchers
Data available in the cloud
Unique in working from unit tests not specifications
Open sourced data available for analysis
38 contests, 40,000 users

Blogs and Sites
April 29, 2015
May 15, 2014
www.codehunt.com
research.microsoft.com/codehunt
research.microsoft.com/codehuntcommunity
Data site on Github
Aim: Involve Students

**Enjoyment** adds to long term retention on a task

**Discovery** is a powerful driver, contrasting with direct instructions

**Gaming** joins these two, and is hugely popular

Can we add these elements to coding?

**Code Hunt** can!

www.codehunt.com
It’s a game!

1. iterative gameplay
2. adaptive
3. personalized
4. no cheating
5. clear winning criterion

Score is based on

• how many puzzles solved,
• how well solved, and
• when solved
Code Hunt

• Is a serious programming game
• Works in C# and Java (Python coming)
• Appeals to coders wishing to hone their programming skills
• And also to students learning to code
• Code Hunt has had over 450,000 users in a year with around 1,000 users a day
• Stickiness (loyalty) is very high
SELECT SECTOR

SECTOR 01

[Games/Code Hunt]
Discover the arithmetic operations applied to 'x'.

```csharp
using System;
public class Program {
    public static int Puzzle(int x) {
        if (x==1) return 3;
        return (x*2);
    }
}
```
You repaired and captured the code fragment.

**SKILL RATING:**

you wrote elegant code!

**TOTAL SCORE: 18**
The ah-ha moment

<table>
<thead>
<tr>
<th>2,353 players</th>
<th>41.0 average tries per level</th>
</tr>
</thead>
<tbody>
<tr>
<td>350 top players</td>
<td>7.6 average tries per level</td>
</tr>
</tbody>
</table>

Code Hunt can identify top coders
Lesson 6: Following the Data

• Java is provided by a source-to-source translator
  • We watched which features players used and what errors they made to concentrate translation efforts for maximum effect

• The bank of over 400 puzzles records difficulty levels
  • These are updated by crowdsourcing users attempts

• The vast number of attempts at solving puzzles gives reliable data as to where programmers have difficulty – see open sourced data

For ImCupSept
257 users x 24 puzzles x approx. 10 tries = about 13,000 programs`
Summary

• Pex → practice impact by surrounding, retargeting, simplifying
  • Moles/Fakes, Pex4Fun/Code Hunt, IntelliTest

• Lessons in transferring tools to practice
  1. Evolving vision
  2. Landing your first customer
  3. Human factors
  4. Misconceptions
  5. Embracing feedback

• Collaboration/engagement with academia

• Educational impact and lesson learned
  6. Following the data

http://research.microsoft.com/pex/

http://www.codehunt.com/
Thanks for attending this ACM SIGSOFT Webinar!

Information on future and past SIGSOFT webinars is found at:

http://www.sigsoft.org/resources/webinars.html
Appendix
Websites

Game  www.codehunt.com
Project research.microsoft.com/codehunt
Community research.microsoft.com/codehuntcommunity
Data Release github.com/microsoft/code-hunt
Blogs Linked on the Project page
Office Mix mix.office.com
Results and rewards

Detailed results are presented

Ranking is always based on score, not on time or attempts

Score is based on

• how many puzzles solved,
• how well solved, and
• when solved

Special rewards for contests
Comparing Code Hunt

Code Hunt

A serious online coding game
Built on symbolic execution
Addressing various audiences
Data available in the cloud

IPCC
Annual programming contest
Scored by unit testing
Addressing university students
Questions available on the web