Teaching High Volume Automated Testing (HiVAT) via Interactive Gaming

Tao Xie
North Carolina State University

In collaboration with Nikolai Tillmann, Peli de Halleux@Microsoft Research
Levels of Test Oracles

• Expected output for an individual test input
  – In the form of **assertions in test code**

• Properties applicable for multiple test inputs
  – **Crash** (uncaught exceptions) or not, related to robustness issues, supported by most tools
  – **Properties in production code**: Design by Contract (precondition, postcondition, class invariants) supported by Parasoft Jtest, Google CodePro AnalytiX
  – **Properties in test code**: Parameterized unit tests supported by MSR Pex, AgitarOne

http://people.engr.ncsu.edu/txie/publications.htm#ac12-devtest
Economics of Test Oracles

• Expected output for an individual test input
  – Easy to manually verify for one test input
  – Expensive/infeasible to verify for many test inputs
  – Limited benefits: only for one test input

• Properties applicable for multiple test inputs
  – Not easy to write (need abstraction skills)
  – But once written, broad benefits for multiple test inputs
Assert behavior of multiple test inputs
Design by Contract

- Example tools: Parasoft Jtest, Google CodePro AnalytiX, MSR Code Contracts, MSR Pex
- **Class invariant**: properties being satisfied by an object (in a consistent state) [AgitarOne allows a class invariant helper method used as test oracles]
- **Precondition**: conditions to be satisfied (on receiver object and arguments) before a method can be invoked
- **Postcondition**: properties being satisfied (on receiver object and return) after the method has returned
- Other types of specs also exist

public virtual int Add(object value) {
    Contract.Requires(value != null);
    Contract.Ensure(Count == Contract.OldValue(Count) + 1);
    Contract.Ensure(Contract.Result<int>() == Contract.OldValue(Count));

    if (count == items.Length) EnsureCapacity(count + 1);
    items[count] = value;
    return count++;
}

[ContractInvariantMethod]
void ObjectInvariant() {
    Contract.Invariant(items != null);
}
Parameterized Unit Testing

- **Parameterized Unit Test** = Unit Test with *Parameters*
- Separation of concerns
  - Data is generated by a tool
  - Developer can focus on functional specification

```csharp
void TestAdd(List list, int item) {
    Assume.IsTrue(list != null);
    var count = list.Count;
    list.Add(item);
    Assert.AreEqual(count + 1, list.Count);
}
```

Parameterized Unit Tests are Algebraic Specifications

- A Parameterized Unit Test can be read as a universally quantified, conditional axiom.

```csharp
void TestReadWrite(Res r, string name, string data) {
    Assume.IsTrue(r!=null & name!=null && data!=null);
    r.WriteResource(name, data);
    Assert.AreEqual(r.ReadResource(name), data);
}

∀ string name, string data, Res r:
    r ≠ null ∧ name ≠ null ∧ data ≠ null ⇒
    equals(
        ReadResource(WriteResource(r, name, data).state, name),
        data)
```
Parameterized Unit Tests in Pex
Parameterized Unit Tests (PUTs) commonly supported by various test frameworks

- .NET: Supported by .NET test frameworks
  - http://www.mbunit.com/
  - http://www.nunit.org/
  - ...

- Java: Supported by JUnit 4.X
  - http://www.junit.org/

Generating test inputs for PUTs supported by tools

- .NET: Supported by Microsoft Research Pex
  - http://research.microsoft.com/Pex/

- Java: Supported by Agitar AgitarOne
  - http://www.agitar.com/
http://people.engr.ncsu.edu/txie/publications/etx07.pdf
Automated Test Generation

- Recent advanced technique: Dynamic Symbolic Execution/Concolic Testing
  - **Instrument** code to **explore** feasible paths

- Example tool: Pex from Microsoft Research (for .NET programs)

P. Godefroid, N. Klarlund, and K. Sen. **DART: directed automated random testing.** In *Proc. PLDI 2005*
K. Sen, D. Marinov, and G. Agha. **CUTE: a concolic unit testing engine for C.** In *Proc. ESEC/FSE 2005*
N. Tillmann and J. de Halleux. **Pex - White Box Test Generation for .NET.** In *Proc. TAP 2008*
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");
}
```

<table>
<thead>
<tr>
<th>Constraints to solve</th>
<th>Data</th>
<th>Observed constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>a==null</td>
<td>null</td>
<td>a==null</td>
</tr>
<tr>
<td>a!=null &amp;&amp; !(a.Length&gt;0)</td>
<td>{}</td>
<td>a!=null &amp;&amp; !(a.Length&gt;0)</td>
</tr>
<tr>
<td>a!=null &amp;&amp; a.Length&gt;0 &amp;&amp; a[0]!=1234567890</td>
<td>{123...}</td>
<td>a!=null &amp;&amp; a.Length&gt;0 &amp;&amp; a[0]!=1234567890</td>
</tr>
<tr>
<td>a!=null &amp;&amp; a.Length&gt;0 &amp;&amp; a[0]==1234567890</td>
<td>{}</td>
<td>a!=null &amp;&amp; a.Length&gt;0 &amp;&amp; a[0]==1234567890</td>
</tr>
</tbody>
</table>

Done: There is no path left.
Automating Test Generation

- **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]

- **Loops**
  - **Fitnex** [Xie et al. DSN 09]

- **Code evolution**
  - eXpress [Taneja et al. ISSTA 11]

http://people.engr.ncsu.edu/txie/publications.htm
Pex on MSDN DevLabs Incubation Project for Visual Studio

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

Academic: 17,366
DevLabs: 13,022
Total: 30,388

http://research.microsoft.com/projects/pex/
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.
Parameterized Unit Test = Unit Test with Parameters

Separation of concerns
- Data is generated by a tool
- Developer can focus on functional specification

```csharp
void TestAdd(List list, int item) {
    Assume.IsTrue(list != null);
    var count = list.Count;
    list.Add(item);
    Assert.AreEqual(count + 1, list.Count);
}
```
Teaching HiVAT: Test Generation == Human Assistance to Test Generation
Challenges Faced by Test Generation Tools

- Example: Dynamic Symbolic Execution/Concolic Testing
  - **Instrument** code to **explore** feasible paths
  - Challenge: path explosion

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

Total block coverage achieved is **50%**, lowest coverage **16%**.

- **object-creation problems (OCP)** - **65%**
- **external-method call problems (EMCP)** – **27%**
A graph example from QuickGraph library

- **Class Graph**:
  ```csharp
class Graph : IVEListGraph { …
    public void AddVertex (IVertex v) {
        vertices.Add(v); // B1
    }

    public Edge AddEdge (IVertex v1, IVertex v2) {
        if (!vertices.Contains(v1))
            throw new VNotFoundException('""'); // B2
        if (!vertices.Contains(v2))
            throw new VNotFoundException('""'); // B3
        Edge e = new Edge(v1, v2);
        edges.Add(e); } }
```

- **Class DFSAlgorithm**: (DFS: Depth First Search)
  ```csharp
class DFSAlgorithm { …
    public void Compute (IVertex s) {
        if (graph.GetEdges().Size() > 0) { // B4
            isComputed = true;
            foreach (Edge e in graph.GetEdges()) {
                … // B5
            }
        }
```
Example Object-Creation Problem

```csharp
00: class Graph : IVEListGraph { ...
03:   public void AddVertex (IVertex v) {
04:       vertices.Add(v); // B1
06:   public Edge AddEdge (IVertex v1, IVertex v2) {
07:       if (!vertices.Contains(v1))
08:           throw new VNotFoundException(""");
09:       // B2
10:       if (!vertices.Contains(v2))
11:           throw new VNotFoundException(""");
12:       // B3
14:       Edge e = new Edge(v1, v2);
15:       edges.Add(e); } }

//DFS:DepthFirstSearch
18: class DFSAlgorithm { ...
23:   public void Compute (IVertex s) { ...
24:       if (graph.GetEdges().Size() > 0) { // B4
25:           isComputed = true;
26:           foreach (Edge e in graph.GetEdges()) {
27:               ... // B5
28:           }
29:       }
```
Challenges Faced by Test Generation Tools

- Example: Dynamic Symbolic Execution/Concolic (Pex)
  - **Instrument** code to **explore** feasible paths
  - Challenge: path explosion

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

Total block coverage achieved is **50%**, lowest coverage **16%**.

- **Object-creation problems (OCP) - 65%**
- **External-method call problems (EMCP) – 27%**
Example External-Method Call Problems (EMCP)

- **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:** `String.Format` do not cause any problem

```
static string GetDefaultConfigFile(string assemblyFile) {
    string configFilename = assemblyFile + ".config";
    if (File.Exists(configFilename)) {
        return configFilename;
    }
    return null;
}
```

```
public ExecutorWrapper(string assemblyFilename, ...) {
    assemblyFilename = Path.GetFullPath(assemblyFilename);
}
```

```
public AssertActualExpectedException {
    this.actual += String.Format("(0)",
        actual.GetType().FullName);
    this.expected += String.Format("(0)",
        expected.GetType().FullName);
}
```

Figure 1: Three simplified methods from xUnit
Tackle object-creation problems with **Factory Methods**
Human Can Help! External-Method Call Problems (EMCP)

Tackle external-method call problems with **Mock Methods** or **Method Instrumentation**

Mocking `System.IO.File.ReadAllText`

```csharp
[Test, Moled]
[Description("This 'mocks' the System.IO.File class with a custom delegate")]
public void ReadFileContentWithMoles()
{
    // arrange ('mock' the FileSystem with a delegate)
    System.IO.Moles.MFile.ReadAllTextString =
    (fname => fname == FileName ? FileContent : "WrongFileName");

    // act
    var testTarget = new TestTarget.TestTarget();
    testTarget.ReadFileContent(FileName);

    // assert
    Assert.AreEqual(FileContent, testTarget.FileContent);
}
```
Tools typically don’t communicate challenges faced by them to enable *cooperation between tools and users*.

We typically don’t teach students how to cooperate with tools.

CSC 712 Software Testing and Reliability (graduate)

- Fall 2010, Fall 2009, Fall 2008: ~20 students
- [http://research.csc.ncsu.edu/ase/courses/csc712/2008fall/](http://research.csc.ncsu.edu/ase/courses/csc712/2008fall/)
- [http://sites.google.com/site/teachpex/](http://sites.google.com/site/teachpex/)

Weekly Schedule

- Mondays: coverage criteria
  - Ammann/Offutt text book
- Weds: hands-on skills
  - Microsoft Research tools: Pex, Code Contract, CHESS, NModel

Course Assignments

- **Warm-up HW**
  - Given ShoppingCart in C#, write C# unit tests based on given Java unit tests
  - Write PUTs to achieve full code coverage and show behavioral differences of Java and C# versions

- **HW on Test Generalization**
  - A bounded unique element int stack
  - ShoppingCart

- **Project (on an open source project)**
  - Midterm report: Test generalization
  - Final report: Write new PUTs

http://people.engr.ncsu.edu/txie/publications/fase11-put.pdf
Email feedback from a Microsoft Academic Relations Manager

“I was just talking to a colleague of mine on the Windows team who just visited NCSU – she was blown away by the knowledge of students from your testing class – you are doing some amazing work!”
Microsoft Research Pex for Fun
Teaching and Learning CS via Social Gaming

www.pexforfun.com

The contributed concept of Coding Duel games as major game type of Pex for Fun since Summer 2010

### Behind the Scene of Pex for Fun

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

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

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

#### Partial Table
<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>

#### Observation
- **Secret Impl** behaves identically to **Player Impl**.
Fun, Educational

- Coding duels
- Brain exercising/learning while having fun
- Fun: iterative, adaptive/personalized with win criterion
- Abstraction/generalization, debugging, problem solving

Course Description: This is a complementary course that includes exercises for selected materials for C# from Sharp Kids: http://msdn.microsoft.com/en-us/bb308756.aspx. Questions and feedback are welcome.

Teacher: TaoXie

Associated Pages:
- CS4Kids Statements
- CS4Kids Code Blocks and Indenting Your Code
- CS4Kids Variables
- CS4Kids Operators
- CS4Kids Converting Between Types
- CS4Kids Branching
- CS4Kids Looping
- CS4Kids The For Loop
- CS4Kids The While Loop
- CS4Kids Whole Program Structure
- CS4Kids Using Class Libraries

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

Registered Students:
Coding Duel Competition
@ICSE 2011

«ICSELevel2Challenge10» - Can you fill the puzzle method to match the secret computation? - you already made 13 attempts

Coding Duels

| #0 | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | #11 | #12 | #13 | #14 | #15 | #16 | #17 | #18 | #19 | #20 | #21 | #22 | #23 | #24 | #25 | #26 | #27 | #28 | #29 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| TaoXia | 3 | 2 | 3 | 3 | 1 | 2 | 6 | 6 | 1 | 2 | 4 | 13 | 3 | 2 | 24 |
| FokkIt1 | 7 | 4 | 4 | 1 | 2 | 12 | 17 | 22 | 25 | 10 | 26 | 7 | 2 | 1 | 3 |
| RPortoAcreu | 4 | 3 | 4 | 8 | 12 | 33 | 40 | 31 | 17 | 112 | 70 | 161 | 2 | 4 | 7 | 8 | 13 | 13 | 2 | 11 | 8 | 21 | 10 | 6 | 17 | 23 | 63 |
| jmaras | 4 | 2 | 3 | 4 | 5 | 5 | 8 | 10 | 49 | 8 | 14 | 6 | 3 | 17 | 4 | 2 | 4 | 9 | 10 | 2 | 34 | 8 | 3 |
| chenfun | 3 | 2 | 5 | 5 | 7 | 6 | 2 | 1 | 3 | 18 | 1 |
| cdragert | 4 | 5 | 4 | 6 | 8 | 3 | 24 | 15 | 22 | 18 | 4 | 18 | 8 | 180 | 10 | 2 | 50 | 3 | 3 | 2 | 5 | 2 | 5 | 4 | 8 | 16 | 7 | 9 | 10 | 21 |
| jamsheicht | 4 |
| JMacFan | 2 | 2 | 18 | 12 | 6 | 18 |
| MEllies | 11 | 2 | 4 | 13 | 14 | 18 | 17 | 9 | 1 |
| shauvik | 2 | 5 | 2 | 3 | 5 | 11 | 6 |
| almancos | 4 | 2 | 4 | 3 | 5 | 10 | 4 | 1 | 3 | 2 | 2 | 1 | 5 | 7 | 22 | 5 | 3 | 5 | 3 | 15 | 16 | 21 | 8 | 12 | 22 | 1 |
| SCLSUPE | 5 | 3 | 4 | 7 | 7 | 21 | 13 | 5 | 3 | 14 | 4 | 2 | 5 | 6 | 5 | 6 | 3 | 4 | 3 | 0 | 15 | 3 | 11 | 15 | 10 | 7 |
| thefour | 6 | 4 | 3 | 12 | 15 | 6 | 15 | 21 | 7 | 15 | 28 | 2 | 7 | 4 | 2 | 4 | 14 | 5 | 7 | 9 | 8 | 2 | 3 | 5 | 14 | 6 | 13 | 7 | 14 | 62 |
| anchii | 6 | 4 | 3 | 12 | 15 | 6 | 15 | 21 | 7 | 15 | 28 | 2 | 7 | 4 | 2 | 4 | 14 | 5 | 7 | 9 | 8 | 2 | 3 | 5 | 14 | 6 | 13 | 7 | 14 | 62 |
| malleres | 4 | 2 | 3 | 6 | 5 | 23 | 18 | 30 | 10 | 17 | 36 | 20 | 224 | 2 | 4 | 12 | 5 | 9 | 2 | 4 | 8 | 9 | 1 | 2 | 18 | 15 | 3 | 17 | 29 | 79 |
| fi4u | 4 | 2 | 4 | 5 | 7 | 10 | 22 | 11 | 9 | 4 | 12 | 12 | 7 | 11 | 16 | 8 | 6 | 2 | 1 |
| kwic | 7 | 6 | 8 | 3 | 7 | 7 |
| MIKAND | 2 | 2 | 1 | 5 | 8 | 9 | 17 | 7 | 5 | 9 | 7 | 2 | 9 | 4 | 2 | 4 | 4 | 5 | 5 | 2 | 5 | 5 | 4 | 13 | 10 |
| ambora | 6 | 7 | 6 | 14 | 13 | 22 | 26 | 51 | 58 | 11 | 147 | 96 | 61 | 226 | 23 | 3 | 16 | 13 | 21 | 7 | 3 | 11 | 21 | 3 | 7 | 21 | 26 | 27 | 22 |
| Benny | 4 | 5 | 4 | 13 | 6 | 10 | 13 | 22 | 32 | 53 | 31 | 6 | 9 | 2 | 2 | 2 | 7 | 9 | 5 | 2 | 2 | 4 | 7 | 8 | 4 | 14 | 22 | 21 | 11 | 20 |
| TheRama | 3 | 2 | 3 | 3 | 18 | 4 | 20 | 9 | 17 | 10 | 37 | 2 | 9 | 15 | 5 | 12 | 6 | 2 | 7 | 6 | 3 | 5 | 11 | 6 | 8 | 12 |
| madking | 2 | 2 | 5 | 11 | 2 | 8 | 13 | 15 | 8 | 10 | 4 | 38 | 4 | 2 | 10 | 6 | 2 | 2 | 7 | 4 | 5 | 7 | 3 | 5 | 2 | 15 |
| nipun | 19 | 9 | 5 | 10 | 2 | 12 | 12 | 12 | 8 | 3 | 3 | 10 | 6 | 6 | 4 | 15 | 24 |
| (no nickname) | 3 | 7 | 2 | 3 |
| eijiadachi | 6 | 3 | 2 | 4 | 7 | 13 | 17 | 11 | 9 | 2 | 3 | 3 | 24 | 1 | 3 | 2 | 3 | 4 | 12 |
| schroeter | 3 | 2 | 3 | 6 | 3 | 3 | 6 | 18 | 4 | 12 |
| Ezno | 3 | 1 | 1 |
| (no nickname) | 3 | 4 | 4 | 11 |

public class Program {
    public static int Puzzle(int x) {
        if (x <= 0) return 0;
        if (x == 1) return 0; //1
        if (x == 2) return 0; //2
        if (x == 3) return 0; //3
        if (x == 4) return 0; //4
        if (x == 5) return 0; //4 //7
        if (x == 6) return 0; //4 //13
        if (x == 7) return 0; //4 //21
        if (x == 8) return 0; //34
        if (x == 9) return 0; //55
        if (x == 10) return 0; //89
        if (x == 11) return 0; //144
        if (x == 12) return 0; //233
        if (x == 72) return (x - 8);
        if (x == 963) return (x - 3);
        if (x == 964) return (x - 4);
        if (x == 965) return (x - 1);
        if (x == 966) return (x - 2);
        if (x == 995) return (x - 3);
        if (x == 996) return (x - 4);
        if (x == 997) return (x - 1);
        if (x == 998) return (x - 2);
        if (x == 999) return (x - 3);
        return 0;
    }
}

using System;
using System.Diagnostics.Contracts;

class Program {
    // param:pref_id:(0-201) Numeric identifier of this preference.
    // param:value:(max. 127 characters) Value of the preference to set. Set it to "0" or "" to remove this preference.
    // Can you write preconditions in Code Contracts for the above natural-language requirements?
    public static int Puzzle(int pref_id, string value)
    {
        return pref_id;
    }
}

http://pexforfun.com/gradsofteng
public static string Puzzle(int[] elems, int capacity, int elem) {
    if ((capacity <= 0) || (elems == null) || (elems.Length > (capacity + 1)))
        return "Assumption Violation!";
    UBIntStack s = new UBIntStack(capacity);
    for (int i = 0; i < elems.Length; i++)
        s.Push(elems[i]);
    int origSize = s.GetNumberOfElements();
    //Please fill in below test scenario on the s stack
    //including necessary assumptions (no additional assertions needed)
    //The lines below include assertions to assert the program behavior
    PexAssert.IsTrue(s.GetNumberOfElements() == origSize + 1);
    PexAssert.IsTrue(s.Top() == elem);
    PexAssert.IsTrue(s.IsMember(elem));
    PexAssert.IsTrue(!s.IsEmpty());
    return "s.GetNumberOfElements():" + s.GetNumberOfElements().ToString() + "; \\
    + "s.Top():" + s.Top().ToString() + "; \\
    + "s.IsMember(elem):" + s.IsMember(elem).ToString() + "; \\
    + "s.IsEmpty():" + s.IsEmpty() + "; "; }
Usage Scenarios of Pex for Fun

- Massive Open Online Courses (MOOC): Challenges
  - Grading, addressed by Pex for Fun
  - Cheating [Open Challenge]
- Course assignments (students/professionals)
  - E.g., intro programming, software engineering
- Student/professional competitions
  - E.g., coding-duel competition at ICSE 2011
- Assessment of testing/programming/problem solving skills for job applicants
  - Not just problem solving final results but also process!
Resources

- Test-generation/testing literature
  - https://sites.google.com/site/asergrp/bibli
- Our publications on testing
  - http://people.engr.ncsu.edu/txie/publications.htm
- Other testing resources
  - http://people.engr.ncsu.edu/txie/resources.html
Thank you

http://www.pexforfun.com/
http://research.microsoft.com/pex
http://sites.google.com/site/teachpex/
http://pexase.codeplex.com/
https://sites.google.com/site/asergrp/
Code Contracts

- http://research.microsoft.com/contracts/
- Library to state preconditions, postconditions, invariants
- Supported by two tools:
  - Static Checker
  - Rewriter: turns Code Contracts into runtime checks
- Pex analyses the runtime checks
  - Contracts act as Test Oracle
- Pex may find counter examples for contracts
- Missing Contracts may be suggested
public class ArrayList {
    private Object[] _items;
    private int _size;
    ...

    [ContractInvariantMethod] // attribute comes with Contracts
    protected void Invariant() {
        Contract.Invariant(this._items != null);
        Contract.Invariant(this._size >= 0);
        Contract.Invariant(this._items.Length >= this._size);
    }
}
Parameterized Test Patterns For Effective Testing with Pex

- 17 patterns
Example Pattern: Roundtrip

- Use receiver or argument value of a method before invoking MUT in Assert

Pattern 2.4/5: Roundtrip

```csharp
[PexMethod]
void ToStringParseRoundtrip(int value) {
    // two-way roundtrip
    string s = value.ToString();
    int parsed = int.Parse(s);
    // assert
    Assert.AreEqual(value, parsed);
}
```
Parameterized Test-Driven Development

- Write/refine Contract as PUT
- Write/refine Code of Implementation
- Run $\text{Pex}$
- Fix-it (with $\text{Pex}$), Debug with generated tests
- Use Generated Tests for Regression
Other Example Topics

- Factory methods
- Parameterized mock objects
- Regression testing
- Using Code Contracts for Design by Contracts
  - [http://research.microsoft.com/contracts/](http://research.microsoft.com/contracts/)

- “Code digging” without writing PUTs (no assertions)
  - Used for regression testing, or
  - Used for output inspection (after test selection)