Improving Automation in Developer Testing:

Test Oracles

Tao Xie
Department of Computer Science
North Carolina State University
http://ase.csc.ncsu.edu/

http://ase.csc.ncsu.edu/autotest/
Software Testing Setup

- Test inputs
- Program
- Outputs

Expected Outputs

Test Oracles

T. Xie. Improving Automation in Developer Testing
Software Testing Problems

- **Faster**: How can tools help developers create and run tests faster?
  - Capture/replay techniques
  - IDE supports for writing test code
Software Testing Problems

- Faster: How can tools help developers create and run tests faster?

- Better Test Inputs: How can tools help generate new better test inputs?
  - Generate method arguments
  - Generate method sequences
Software Testing Problems

- **Faster**: How can tools help developers create and run tests faster?

- **Better Test Inputs**: How can tools help generate new better test inputs?

- **Better Test Oracles**: How can tools help generate better test oracles?
Example Unit Test Case

void addTest() {
    ArrayList a = new ArrayList(1);
    Object o = new Object();
    a.add(o);
    assertTrue(a.get(0) == o);
}

- Appropriate method sequence
- Appropriate primitive argument values
- Appropriate assertions

T. Xie. Improving Automation in Developer Testing
Software Testing Problems

• **Faster:** How can tools help developers create and run tests faster?

• **Better Test Inputs:** How can tools help generate new better test inputs?

• **Better Test Oracles:** How can tools help generate better test oracles?
  - Assert behavior of individual test inputs
  - Assert behavior of multiple test inputs
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, CodePro AnalytiX
  - Properties in test code: parameterized unit tests supported by MSR Pex, AgitarOne
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 individual test inputs

Capture and Assert

• Capture and Assert [Xie ECOOP 06]
  – Phase I: Capture the return values of observer methods
  – Phase II: Create assertions to assert return values
  – Also assert uncaught exception throwing
  – Example tools: Parasoft Jtest, CodePro AnalytiX, AgitarOne

• Also called characterization test in JUnit Factory (by Agitar Software Inc.)
public void test1() {
    Stack s1 = new Stack();
    s1.push(3);
    s1.top();
    s1.isMember(3);
}

public void test1() {
    Stack s1 = new Stack();
    s1.push(3);
    assertEquals(s1.top(), 3);
    assertEquals(s1.isMember(3), true);
}
Capture and Assert: issues

• For regression testing only unless you verify
  – If your current version has a bug, the assertion makes sure your bug exists in a future version!

• Ask developers to verify assertions/exceptions in test code
  – Example tools: Parasoft Jtest and CodePro AnalytiX
Verify Outcomes in Parasoft Jtest
Capture and Assert: issues

- For regression testing only unless you verify
  - If your current version has a bug, the assertion makes sure your bug exists in a future version!

- Ask developer to verify assertions/exceptions in test code

- Challenge: which observer methods to invoke? (i.e., which assertions to add?)
UnitPlus @NCSU
Recommend Assertions to Developers

```
public void testAdd() throws Exception {
    Management mgmt = new Management();
    mgmt.add(new Person("Jane Doe", 20));

    // Assert management operations
    assert mgmt.exists(new Person("Jane Doe", 20));
    assert mgmt.getAge(new Person("Jane Doe", 20)) == 20;
    assert mgmt.getPerson("Jane Doe", 20).getName() == "Jane Doe";
    assert mgmt.getPerson("Jane Doe", 20).getAge() == 20;
    assert mgmt.getPeople().size() > 0;
    assert mgmt.isEmpty() == false;
}
```

[Song, Thummalapenta, and Xie ETX 07]
Assert behavior of multiple test inputs

Design by Contract

• Example tools: Parasoft Jtest, CodePro AnalytiX, Microsoft Research Spec Explorer, 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
Microsoft Research Spec Explorer

```java
public static MapContainer<int, int> pins = new MapContainer<int, int>();
public static MapContainer<int, int> balances = new MapContainer<int, int>();

public static void AddCustomer(int customer, int pin, int balance)
{
    Contracts.Requires(!pins.ContainsKey(customer));
    pins[customer] = pin;
    balances[customer] = balance;
}

public static bool VerifyPIN(int customer, int pin)
{
    return pins.ContainsKey(customer) && pins[customer] == pin;
}

public static int InquireBalance(int customer)
{
    Contracts.Requires(pins.ContainsKey(customer));
    return balances[customer];
}

public static bool TryWithdrawal(int customer, int amount)
{
    Contracts.Requires(pins.ContainsKey(customer));
    if (InquireBalance(customer) >= amount)
    {
        balances[customer] -= amount;
        return true;
    }
    else
        return false;
}
```
Assert behavior of multiple test inputs

Parameterized Unit Tests

- Adding parameters turns unit tests into general specs
  
  ```csharp
  [TestMethod]
  void AddParameterizedTest(ArrayList a, object o) {
    Assume.IsTrue(a != null);
    int len = a.Count;
    a.Add(o);
    Assert.IsTrue(a[len] == o);
  }
  ```

- Read as “forall a, o: the given assertion holds”

- Tool chooses argument values that cover all implementation paths
  
  ```csharp
  // 1. case: existing storage used
  AddParameterizedTest(new ArrayList(1), new object());
  // 2. case: new storage allocated
  AddParameterizedTest(new ArrayList(0), new object());
  ```

Supported by MSR Pex [Tillmann&Schulte FSE 05] and AgitarOne

T. Xie. Improving Automation in Developer Testing - Slide adapted from MSR FSE Group 18
Parameterized Unit Tests in Pex

```csharp
[TestClass, PexClass]
public partial class ChunkerTest {
    [PexTest]
    public void Chunk(string value, int count) {
        Chunker chunker = new Chunker(value, count);
        string result = null;
        for (string chunk = chunker.Next(); chunk != null; chunk =
            result += chunk;
        }

        Assert.AreEqual(value, result);
    }
}

return chunk;
```
Assert behavior of multiple test inputs
Software Agitation in AgitarOne

If an Observation reveals a bug, fix it
If it describes desired behavior, click to create a Test Assertion

- Slide adapted from Agitar Software Inc.
Software Agitation in AgitarOne

Image from http://www.agitar.com/
Test Selection/Abstraction

- Test selection based on
  - code coverage, e.g., Parasoft Jtest
  - new behavior [Hangal & Lam ICSE 02, Xie & Notkin ASE 03, Pacheco & Ernst ECOOP 05, d'Amorim et al. ASE 06, ...]
  - special behavior [Xie & Notkin ISSRE 05, ...]
- Test abstraction for overall behavior [Xie & Notkin ICFEM 04, Dallmeier et al. WODA 06, ...]

\[ r(a(S, e).state).state == a \]
\[ (r(S).state, e).state \]
Conclusion

- Software testing is important and yet costly; needs automation

- **Faster:** help developers create and run tests faster
  - Capture/replay techniques
  - IDE supports for writing test code

- **Better Test Inputs:** help generate new better test inputs
  - Generate method arguments
  - Generate method sequences

- **Better Test Oracles:** help generate better test oracles
  - Assert behavior of individual test inputs
  - Assert behavior of multiple test inputs
Example Industrial Developer Testing Tools

- Microsoft Research Pex (for .NET) [http://research.microsoft.com/Pex/](http://research.microsoft.com/Pex/)
- Microsoft Research Spec Explorer (for .NET) [http://research.microsoft.com/specexplorer/](http://research.microsoft.com/specexplorer/)
Questions?

Contact info:
Tao Xie (xie@csc.ncsu.edu)
Automated Software Engineering Group
North Carolina State University
http://ase.csc.ncsu.edu/