Improving Automation in Developer Testing:

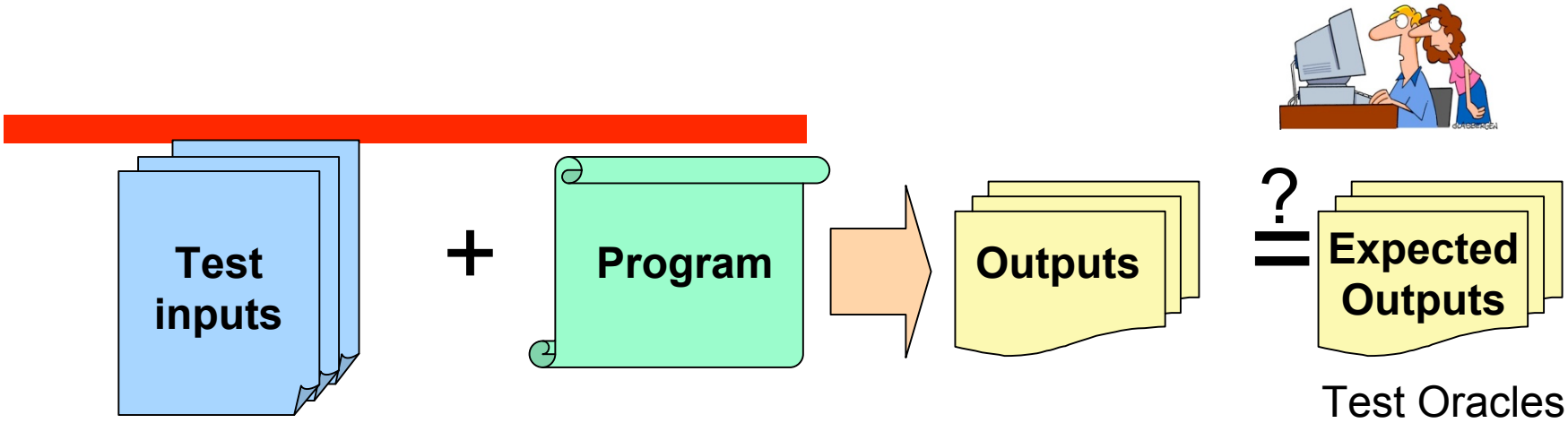
Test Oracles

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

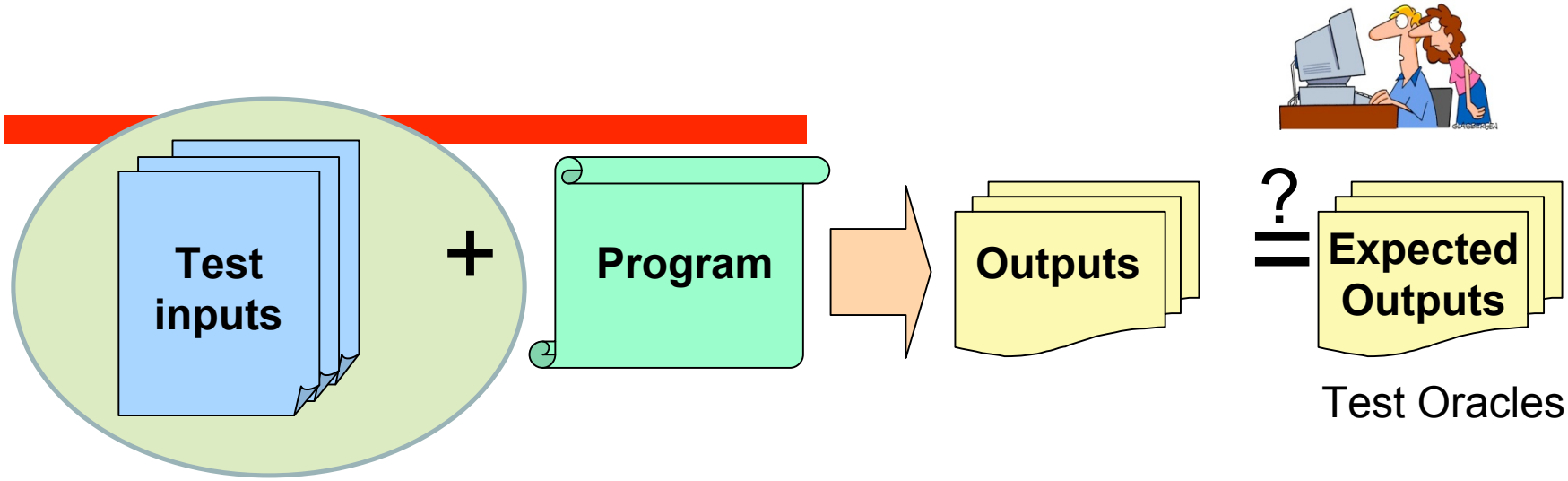
http://ase.csc.ncsu.edu/autotest/

Automated Software Research Group Engineering@NCSU

Software Testing Setup

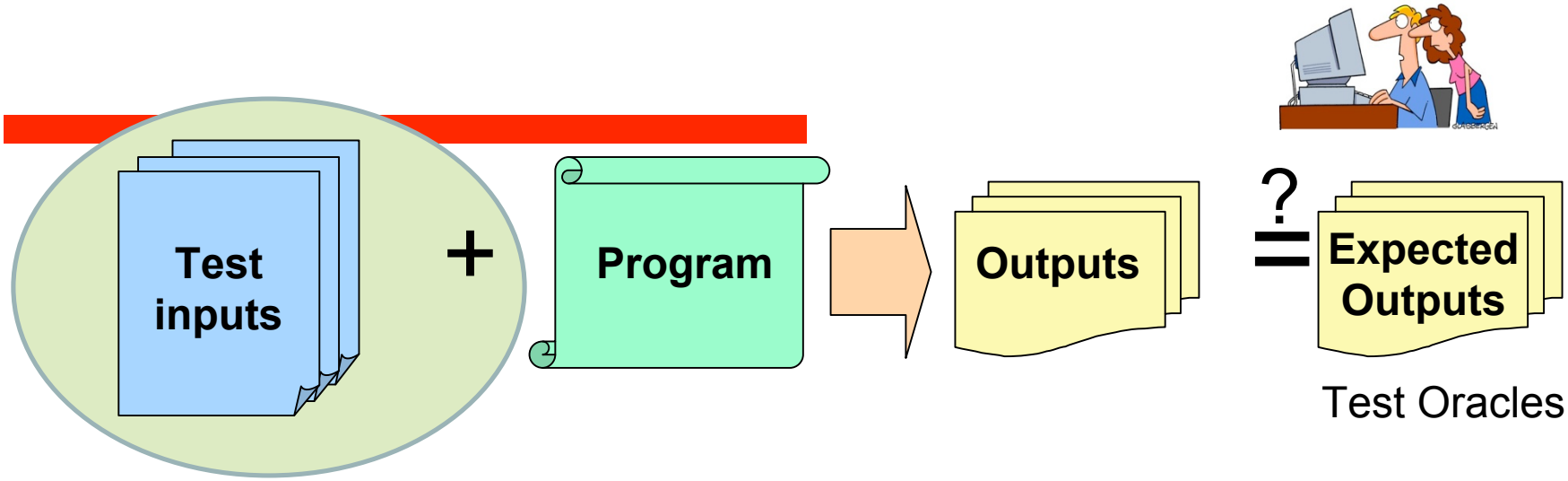


Software Testing Problems



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

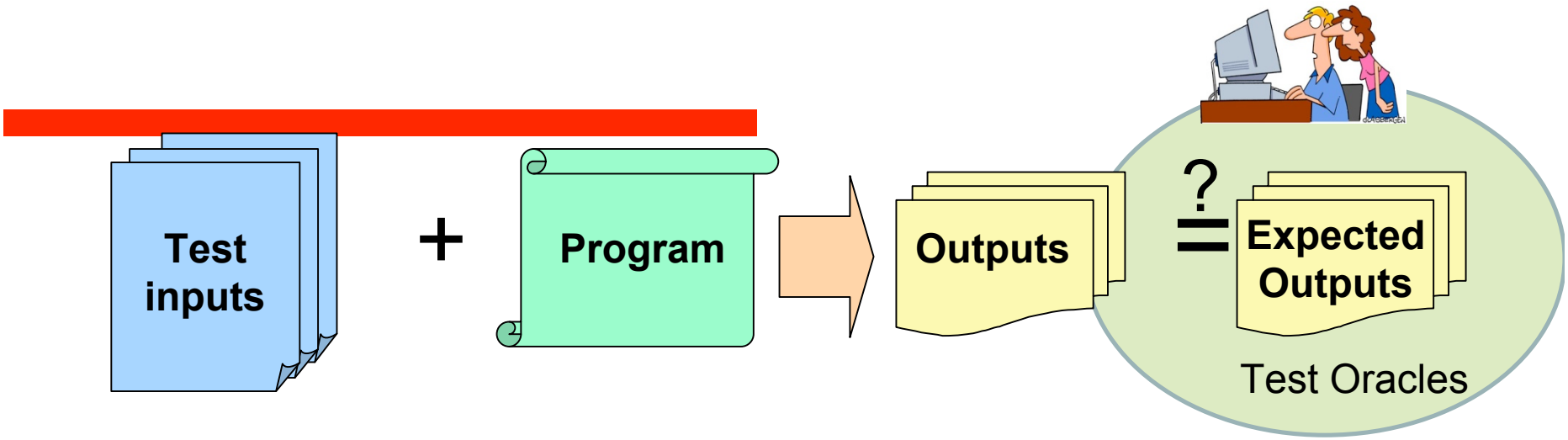
Software Testing Problems



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

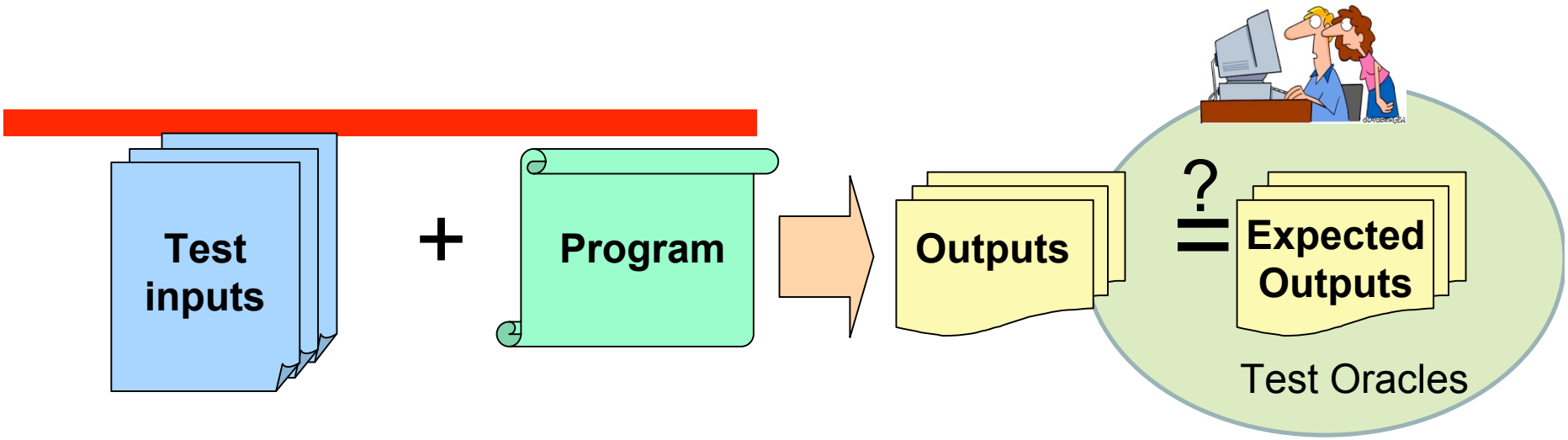
- **Better Test Inputs:** How can tools help generate new better test inputs?
  Generate method arguments
  Generate method sequences

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

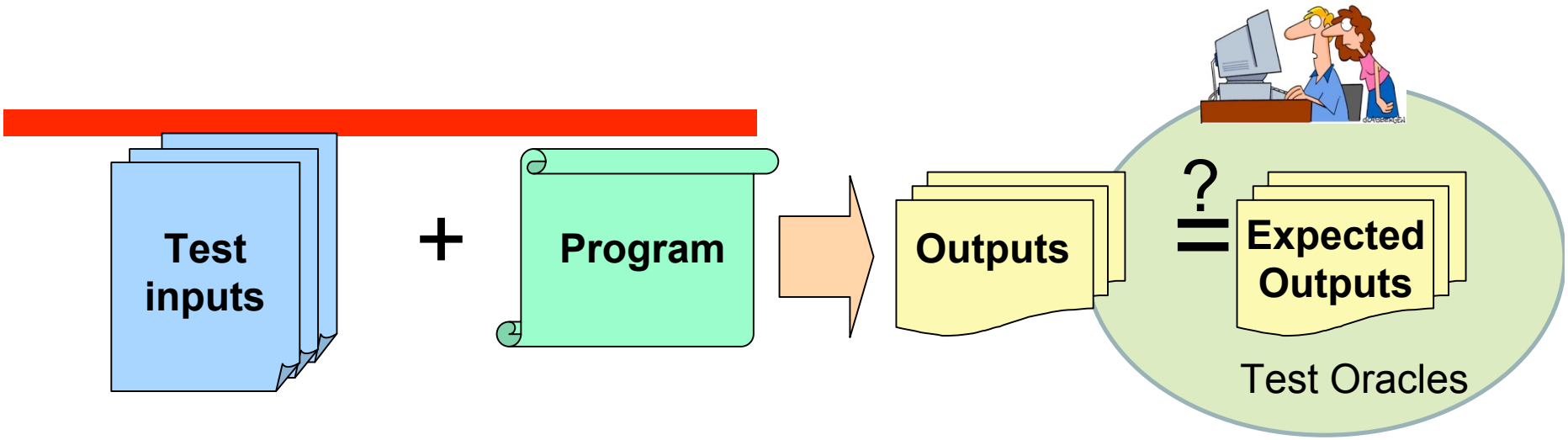



```
void addTest() {
  ArrayList a = new ArrayList(1);
  Object o = new Object();
  a.add(o);
  AssertTrue(a.get(0) == o);
}
```

Test Case = Test Input + Test Oracle

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

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

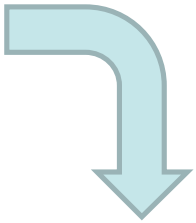
- Properties applicable for multiple test inputs
  - Not easy to write (need abstraction skills)
  - But once written, broad benefits for multiple test inputs

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.)

Capture and Assert: example

```
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

Jtest - BankAccount.java - Parasoft Jtest

File Edit Source Refactor Navigate Search Project Jtest Run Window Help

Package Explorer

- Jtest Examples
  - dbc
  - examples.bank
    - BankAccount.java
    - ChangeOwnership.java
    - Customer.java
    - Deposit.java
    - Statement.java
      - Statement
        - printStatement(BankAccount)
        - Statement()
    - Transaction.java
    - TransactionTest.java
    - Withdraw.java
  - examples.eval
  - examples.initialize
  - examples.inputs
  - examples.leaks
  - examples.queue
  - examples.repository
  - examples.singletons
  - examples.stackmachine
  - examples.suppress
  - examples.unit
  - images
  - JTEST_HOME/jtest.jar - C:\Program Files\Pa
  - JUNIT_JAR - C:\Program Files\ParaSoft\Jtes
  - JRE System Library [1.4]
  - mysql-3.0.10.jar - C:\Program Files\ParaSof
  - build.xml
  - examples.properties
- Jtest Examples.jtest

BankAccount.java

```
35
36      public void apply (Transaction transaction) {
37          if (transaction instanceof Deposit)
38              apply0 ((Deposit) transaction);
39          else if (transaction instanceof Withdraw)
40              apply0 ((Withdraw) transaction);
41          else if (transaction instanceof ChangeOwnership)
42              apply0 ((ChangeOwnership) transaction);
43          balance -= transaction.getTransactionFee ();
44      }
45
46      public void applyInterest() {
```

Testing Tasks | Problems | Coverage | Test Class Outline | Object Repository | Test Case Repository

Testing Tasks (133 tasks)

- [34] Unverified Outcomes
  - [34] Outcome
    - [34] examples.bank
      - [34] BankAccount.java
        - Unverified Outcome: testing BankAccount(int, int): THIS.getBalance() is 0
        - Unverified Outcome: testing BankAccount(int, int): THIS.getBalance() is 7
        - Unverified Outcome: testing BankAccount(int, int): THIS.getCustomerId() is 0
        - Unverified Outcome: testing BankAccount(int, int): THIS.getCustomerId() is 7
        - Unverified Outcome: testing apply(Transaction): THIS.getBalance() is -1
        - Unverified Outcome: testing apply(Transaction): THIS.getBalance() is 0
        - Unverified Outcome: testing apply(Transaction): THIS.getCustomerId() is 0
        - Unverified Outcome: testing apply(Transaction): THIS.getCustomerId() is 7
        - Unverified Outcome: testing applyInterest(): THIS.getBalance() is 0
        - Unverified Outcome: testing applyInterest(): THIS.getBalance() is 7
        - Unverified Outcome: testing applyInterest(): THIS.getCustomerId() is 0
        - Unverified Outcome: testing applyInterest(): THIS.getCustomerId() is 7
        - Unverified Outcome: testing getBalance(): THIS.getCustomerId() is 0
        - Unverified Outcome: testing getBalance(): THIS.getCustomerId() is 7
        - Unverified Outcome: testing getBalance(): returns 0
        - Unverified Outcome: testing getBalance(): returns 7
        - Unverified Outcome: testing getCustomerId(): THIS.getBalance() is 0

Testing Tasks (133 tasks)

Image from http://www.parasoft.com/

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));
}
```

Rename in file (Ctrl+2, R direct access)
- assert org.Management.exists(org.Person)
- assert org.Management.getAge(java.lang.String)
- assert org.Management.howmany()
- assert org.Management.isEmpty()

```
...
Management mgmt = new Management();
mgmt.add(new Person("Jane Doe", 20));
Person person = new Person("Jane Doe", 20);
assertEquals(true, mgmt.exists(person));
}
...
```

State-modifying Method | Observer Methods | Problems | Console | Javadoc | Declaration | Search

Name	Written Fields
org.Management.add(java.lang.St...	[new org.Person.fAge, new org.Person.fName, org.Management.b, org.Manage...
org.Management.add(org.Person)	[org.Management.fCount]

[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

SimpleATM - Microsoft Visual Studio

```
class BankModel
{
    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;
    }
}
```

- Slide adapted from MSR FSE Group

Assert behavior of multiple test inputs
Parameterized Unit Tests

- Adding parameters turns unit tests into general specs

```
[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

```
// 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

Parameterized Unit Tests in Pex

ChunkerDemo - Microsoft Visual Studio

File Edit View Refactor Project Build Debug Data Tools Test Window Community Help

Debug | Any CPU

ChunkerTest_...9_162641_1.cs | ChunkerTest.cs

ChunkerDemo.ChunkerTest | Chunk(string value, int count)

```
            return chunk;
        }
    }

    [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);
        }
    }
}
```

Solution Explorer - Chu...

- Solution 'ChunkerDemo' (1 proj
  - Solution Items
  - ChunkerDemo
    - Properties
    - References
    - ChunkerTest.cs
      - ChunkerTest_7010
      - ChunkerTest_7010
      - ChunkerTest_7010

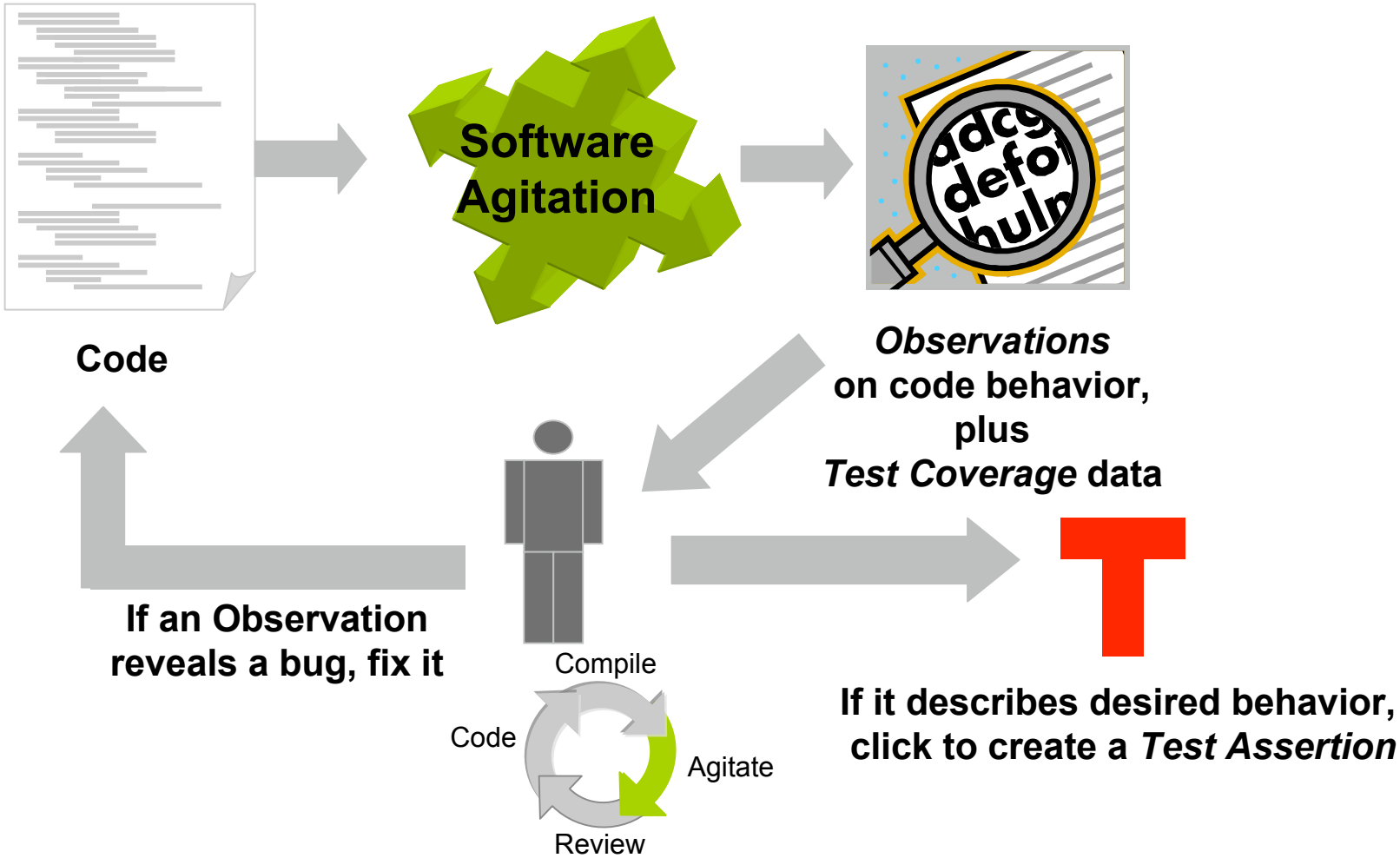
Properties

Code Coverage Results | Output | Code Definition Window | Find Results 1 | Error List | Test Results

Ready | Ln 41 | Col 1 | Ch 1 | INS

Image from http://research.microsoft.com/Pex/

Assert behavior of multiple test inputs
Software Agitation in AgitarOne



Software Agitation in AgitarOne

Agitar - Product.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Agitar Run Window Help

Agitation: agitar-tutorial

Resource | Agitar | Java

Package Explorer | Hierarchy

- agitar-tutorial
  - src
    - tutorial
      - CreditException.java
      - Customer.java
      - Order.java
      - OrderItem.java
      - Product.java
      - Customer.java.ORIG
  - junit.jar
  - JRE System Library [jre1.5.0_08]
  - AGITAR_MOCK_OBJECTS - C:\java
  - AGITAR_TEST_LIB - C:\java\demo

Product.java

```
/**
 * @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 IllegalArgumentExcep
  validateCode(code);
  validateName(name);
  validatePrice(price);
  this.code = code;
  this.name = name;
  this.price = price;
}

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

Outcomes | Outline

- Product
  - <init>(String, String, double) (12
    - NORMAL (42)
    - EXCEPTION (82)
      - IllegalArgumentException
  - getCode() (6)
  - getName() (6)
  - getPrice() (6)
  - setName(String) (56)
  - setPrice(double) (51)
  - toString() (6)
  - *validateCode(String)*
  - *validateName(String)*

Agitar... | Obser... | Snaps... | Factories | Factor... | Agitation | Problems | Experts | Class ... | Cover... | **Console** | JUnit | Stackt...

Outcome: Product -> <init>(String, String, double) -> NORMAL

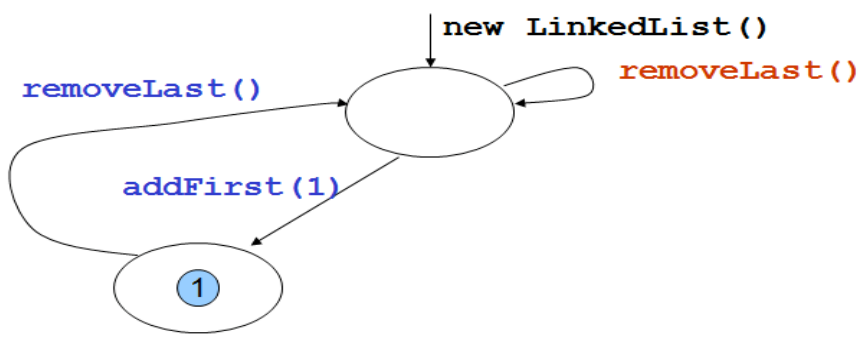
Value		True	False
0.0 <= this.getPrice() <= 1000.0	☑	42	0
this.getCode().length() == 11	☑	42	0
this.getPrice() == price	☑	42	0
this.getName() == name	☑	42	0
this.getCode() == code	☑	42	0
1 <= this.getName().length() <= 20	☐	*42*	*0*
this.getName().length() == name.length()	☐	*42*	*0*

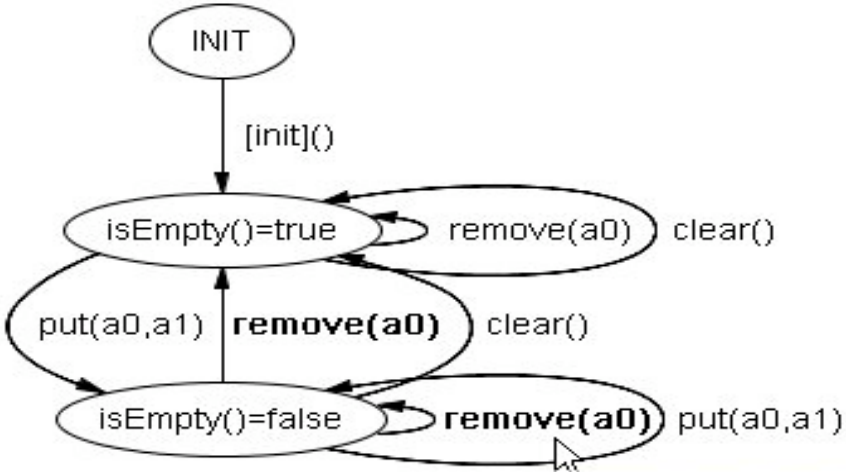
Image from http://www.agitar.com/

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

- Agitar AgitatorOne http://www.agitar.com/
- Parasoft Jtest http://www.parasoft.com/
- CodePro AnalytiX http://www.instantiations.com/
- SilverMark Test Mentor http://www.silvermark.com/

- Microsoft Research Pex (for .NET) http://research.microsoft.com/Pex/
- Microsoft Research Spec Explorer (for .NET) http://research.microsoft.com/specexplorer/
- Avaya Labs Research eXVantage http://www.research.avayalabs.com/

Automated Software Research Group Engineering@NCSU

Questions?

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