Automated Test Generation for AspectJ Programs

Tao Xie[1], Jianjun Zhao [2], Darko Marinov[3], and David Notkin [1]

[1] University of Washington
[2] Fukuoka Institute of Technology
[3] University of Illinois at Urbana-Champaign

Motivation

- AspectJ’s specific constructs require adapting the existing testing concepts, e.g. test-input generation
- Generate tests for AspectJ programs by developing completely new tools
  - Duplicate a large part of the existing Java test-generation tools’ functionality.
- Can we reuse existing tools for Java programs to automatically generate tests for AspectJ programs?
- What research issues to be addressed during the reuse of the existing tools?

Motivation

- AspectJ’s specific constructs require adapting the existing testing concepts, e.g. test-input generation

- Generate tests for AspectJ programs by developing completely new tools
  - Duplicate a large part of the existing Java test-generation tools’ functionality.

- Can we reuse existing tools for Java programs to automatically generate tests for AspectJ programs?

- What research issues to be addressed during the reuse of the existing tools?

Wrasp is proposed to address both questions with wrapper classes, complement **Aspectra** for detecting AspectJ redundant tests [Xie et al. 04]

Straightforward Tool Reuse

- Existing Java test-generation tools (based on bytecode)
  - Parasoft Jtest, NASA Java Pathfinder [Visser et al. ISSTA 04]
    JCrasher [Csallner &Smaragdakis SPE 04], Rostra [Xie et al. ASE 04], Symstra [Xie et al. TACAS 05]
- AspectJ unit testing: testing aspects in isolation
  - Treat a compiled aspect class as the class under test for existing tools
  - Issues: `JionPoint` and `AroundClosure` arguments
- AspectJ integration testing: interaction between base classes and aspects
  - Treat a woven class as the class under test for existing tools

Testing Aspect in Isolation

```
public void testNonNegative1() {
  Stack t0 = new Stack ();
  NonNegative THIS = new NonNegative();
 THIS.ajc$before$NonNegative$1$d9be608f(t0);
}


public void testPushCount1() {
 Stack t0 = new Stack();
  PushCount.ajc$interMethod$PushCount$Stack$
  incrementCount(t0);
}
```

Issues of Integration Testing

- Advice of “call” join points is woven at call sites
  - Dynamic-test-generation tools cannot execute the advice during test generation
  - Indeed, we can weave generated tests together with base classes and aspects (after the tests have been generated)

- Test-weaving compilation may fail when the interfaces of woven classes contain intertype methods
  - Intertype methods don’t appear in base classes’ source

Wrapper Class As Class under Test

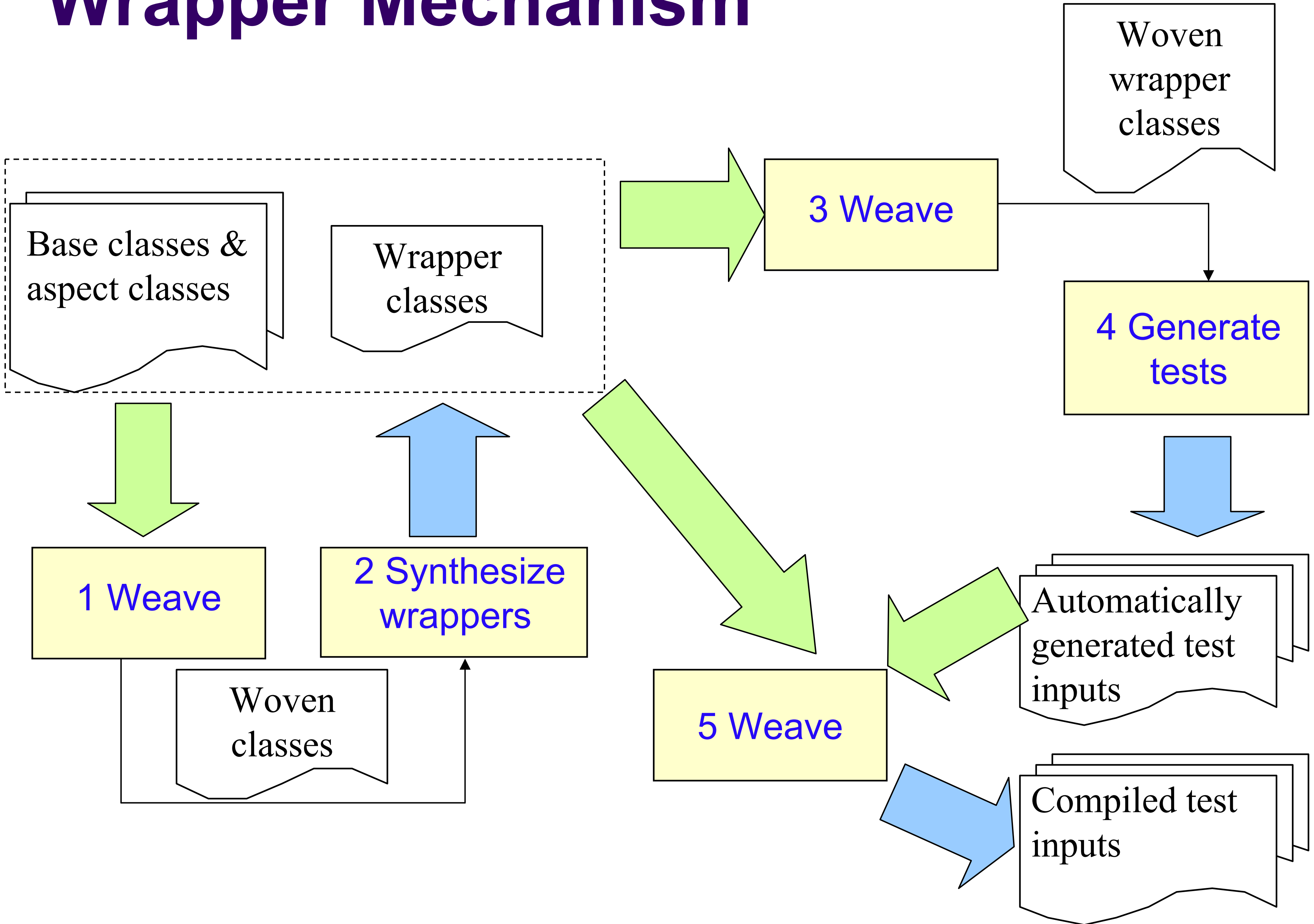
```
public class Stack {
 public Stack()  {…}
 public boolean push(int i){…}
 public int pop()  {…}
}
```

```
aspect PushCount {
 int Stack.count = 0;
 public void Stack.increaseCount(){
    count++;
 }}
```

```
public class StackWrapper {
 Stack s;
 public StackWrapper(){
   s = new Stack();
 }
 public boolean push(int i) {
   return s.push(i);
 }
 public int pop() {return s.pop();}
 public void increaseCount() {
  Class cls = Class.forName("Stack");
  Method meth =
  cls.getMethod("increaseCount",null);
  meth.invoke(s, null);
 }
}
```

- Advice of “call” join points is woven at call sites
- Test-weaving compilation may fail when the interfaces of woven classes contain intertype methods

Wrapper Mechanism



Discussion

- What AOP features make existing test generation tools difficult?
  - Interaction (implementation-based testing fails for missing path)

- What AOP features make existing test generation tools easy?
  - Observable units:
    generate integration tests → detect non-redundant tests for aspects → inspect non-redundant tests [Xie et al. 04]

- What new tools/infrastructures shall the community build?
  - More subjects (beyond http://www.sable.mcgill.ca/benchmarks/)
  - Mutation tools (OO: http://www.ise.gmu.edu/~ofut/mujava/)
  - Coverage measurement tools
  - Typical-fault repository (Non-AOP: U. Nebraska Lincoln)
  - Testing tools specific for AOP features that are not addressed by OO testing tools