

Konrad Jamrozik¹, Gordon Fraser²,
Nikolai Tillman³, Jonathan Halleux³

Augmented Dynamic Symbolic Execution

ASE 2012

¹Saarland University

²University of Sheffield

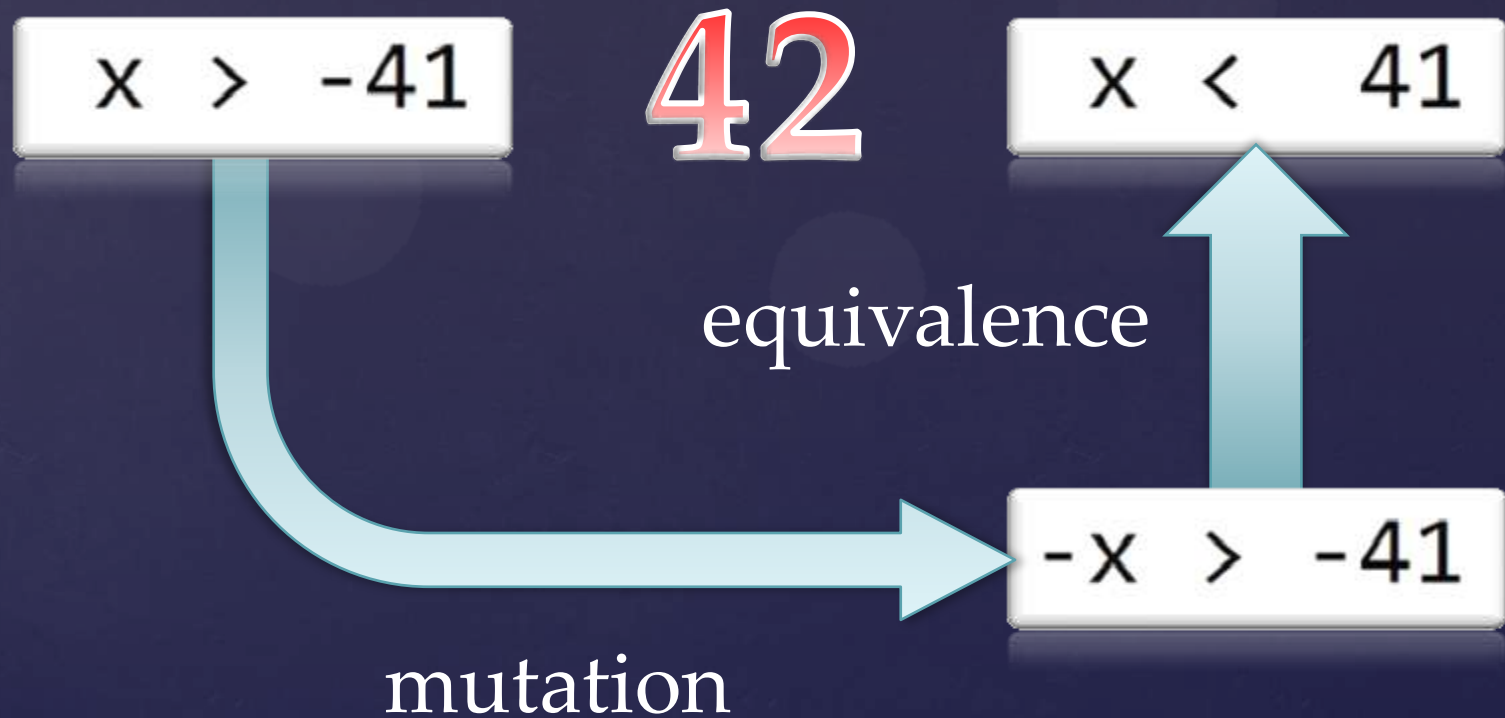
³Microsoft Research

```
public void MethodUnderTest1(int x)
{
    if (10 <= x && x <= 20)
        ComputeInRange();
    else
        ComputeOutOfRange();
}
```

Test inputs generated by DSE: $x = \{0, 10\}$

By ADSE: $x = \{0, 9, 10, 11, 19, 20, 21\}$

To *kill* a mutant, we need a test case that passes on the original program but fails on the mutant.



```
public void MethodUnderTest1(int x)
{
    if (10 <= x && x <= 20)
        Cor < -x range();
    else
        Cor > x+1
        Cor >= x-1 :OfRange();
    == 1
    != 0
}
```

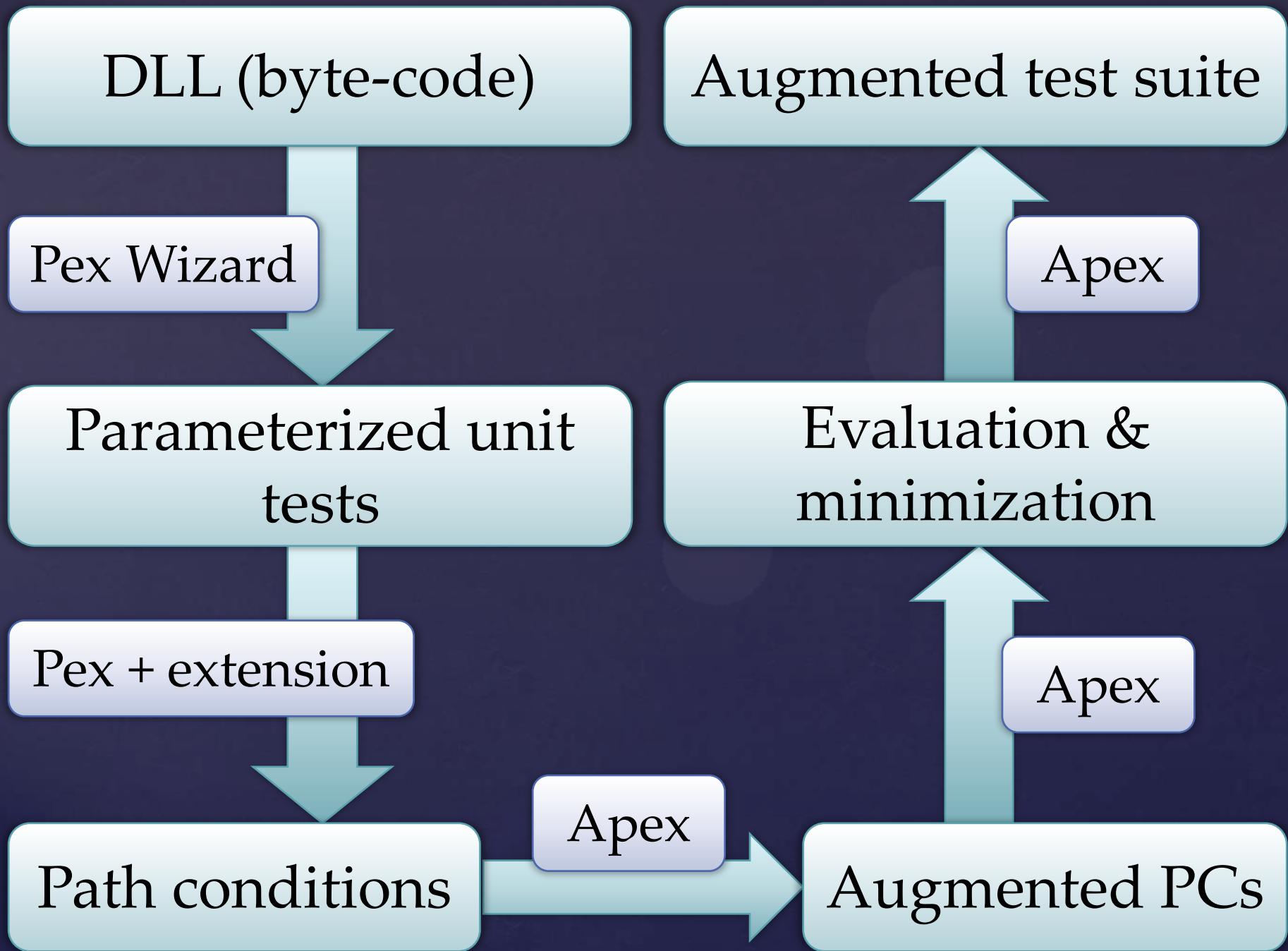
ADSE instances: boundary cases, mutation, logical coverage, exceptions, ...

Tools of the trade

Technology platform: .NET 4.0 / C#

DSE engine: Pex (Program Exploration)

Constraint solver: Z3



Evaluation subjects

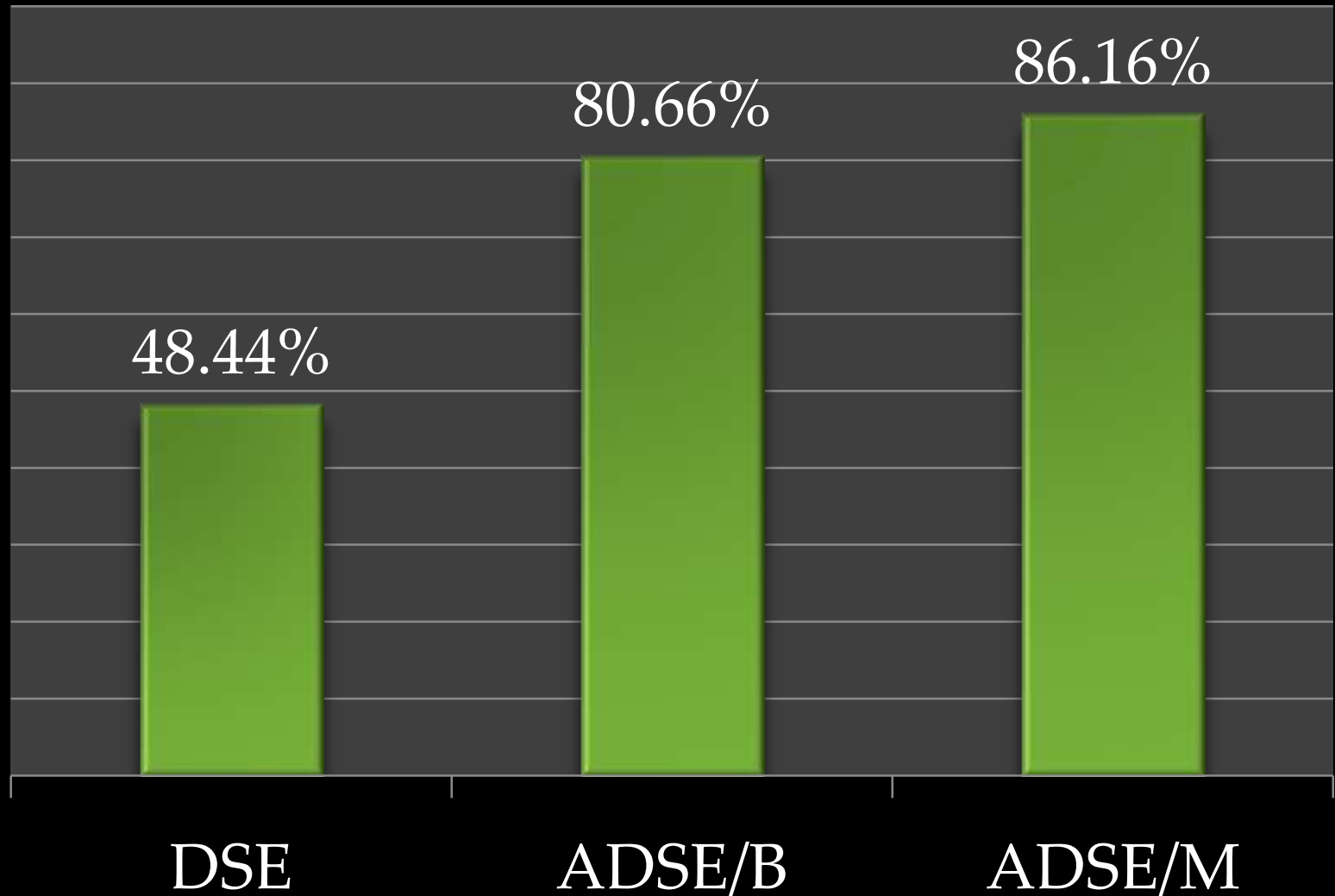
Factorial, Power, MaxValue, Fibonacci, GCD

WBS (Wheel brake system)

FindMiddle, WrapRoundCounter

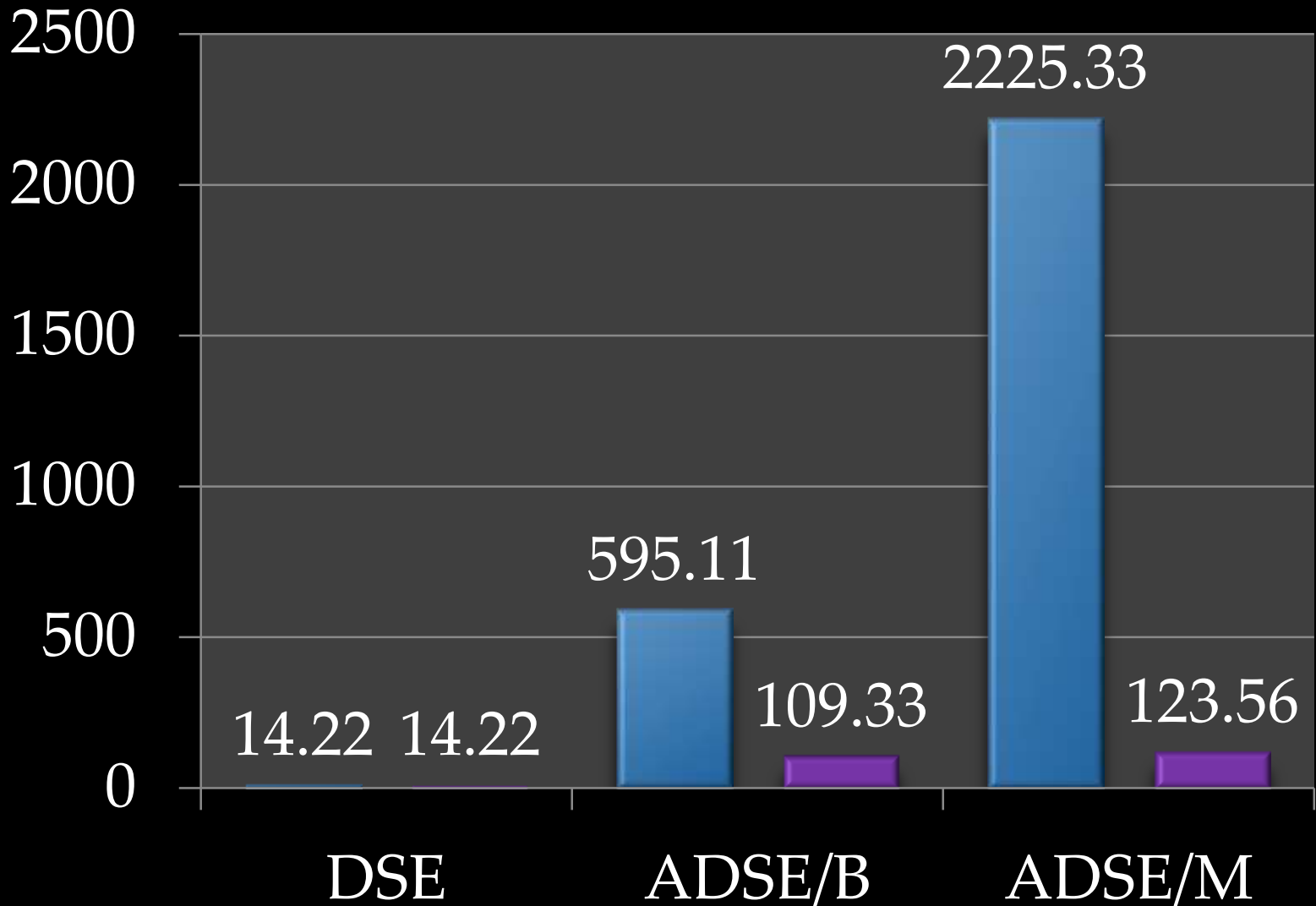
Roops integer examples

■ Mutation Score



■ Path conditions

■ Tests



Boundary ADSE and mutation ADSE significantly increase mutation score as compared to plain DSE, thus significantly increasing resulting test suite defect detection ability.

ADSE can create tests exercising code under test with inputs expected by developers, e.g. covering boundary cases.