An Introduction to Search-based Testing and the EVOSUITE Test Generation Tool

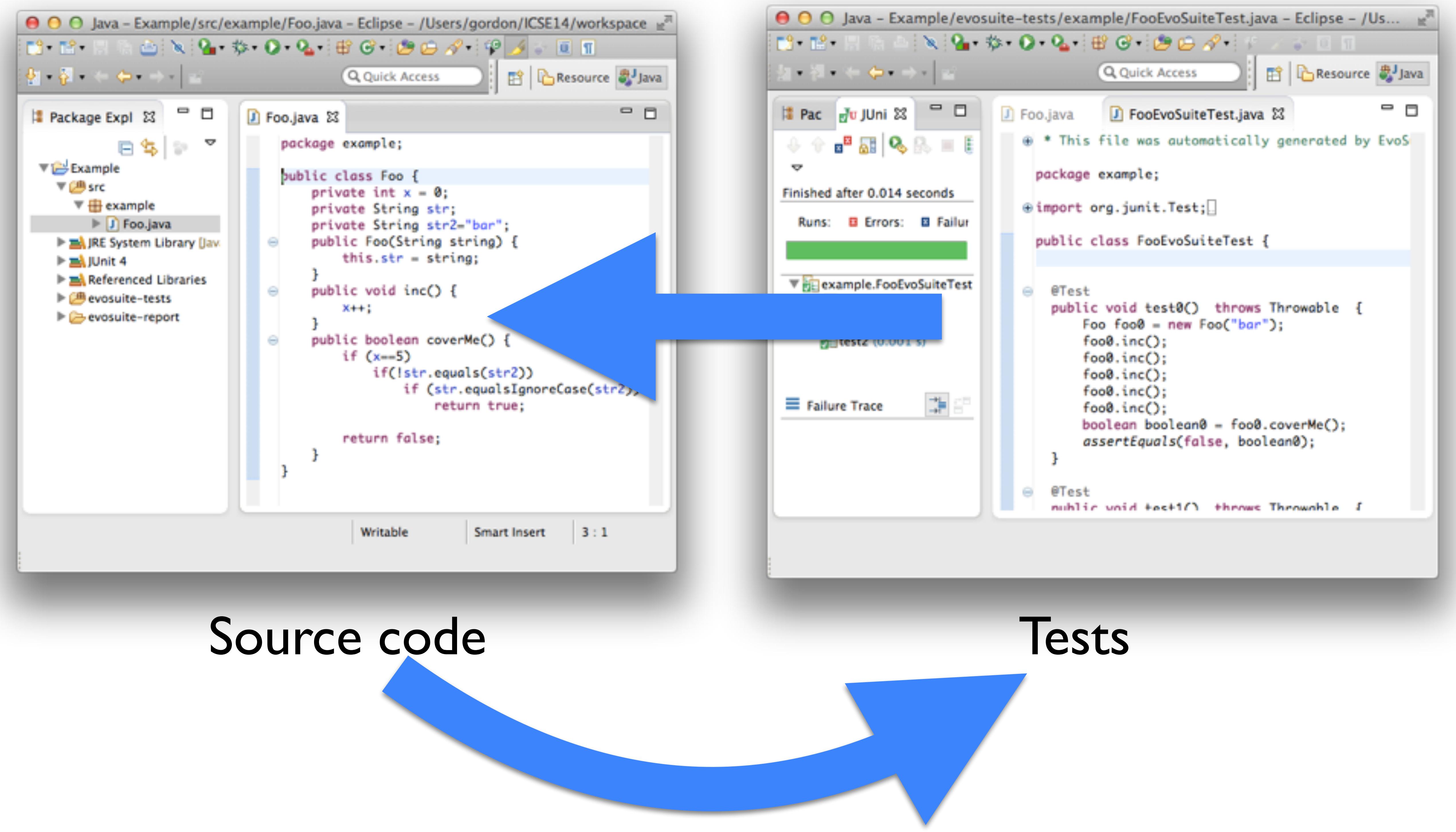
Gordon Fraser, University of Sheffield

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

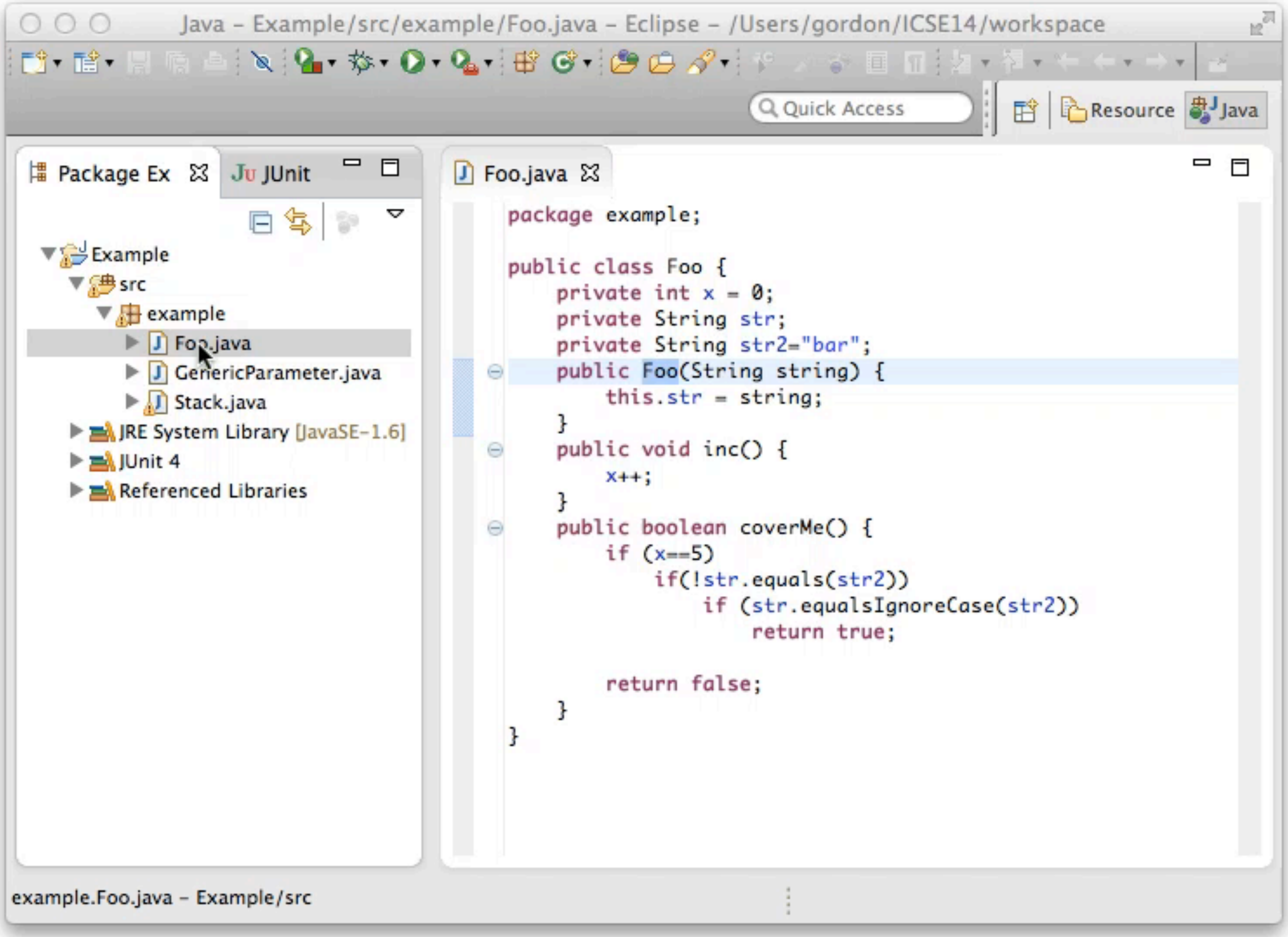

Java – Example/src/example/Foo.java – Eclipse – /Users/gordon/ICSE14/workspace
Package Expl
Example
src
example
Foo.java
JRE System Library
JUnit 4
Referenced Libraries
evosuite-tests
evosuite-report
Writable
Smart Insert
3 : 1
Java – Example/evosuite-tests/example/FooEvoSuiteTest.java – Eclipse – /Us...
Finished after 0.014 seconds
Runs:
Errors:
Failur
example.FooEvoSuiteTest
Failure Trace
Source code
Tests
Automated test generation

Foo.java

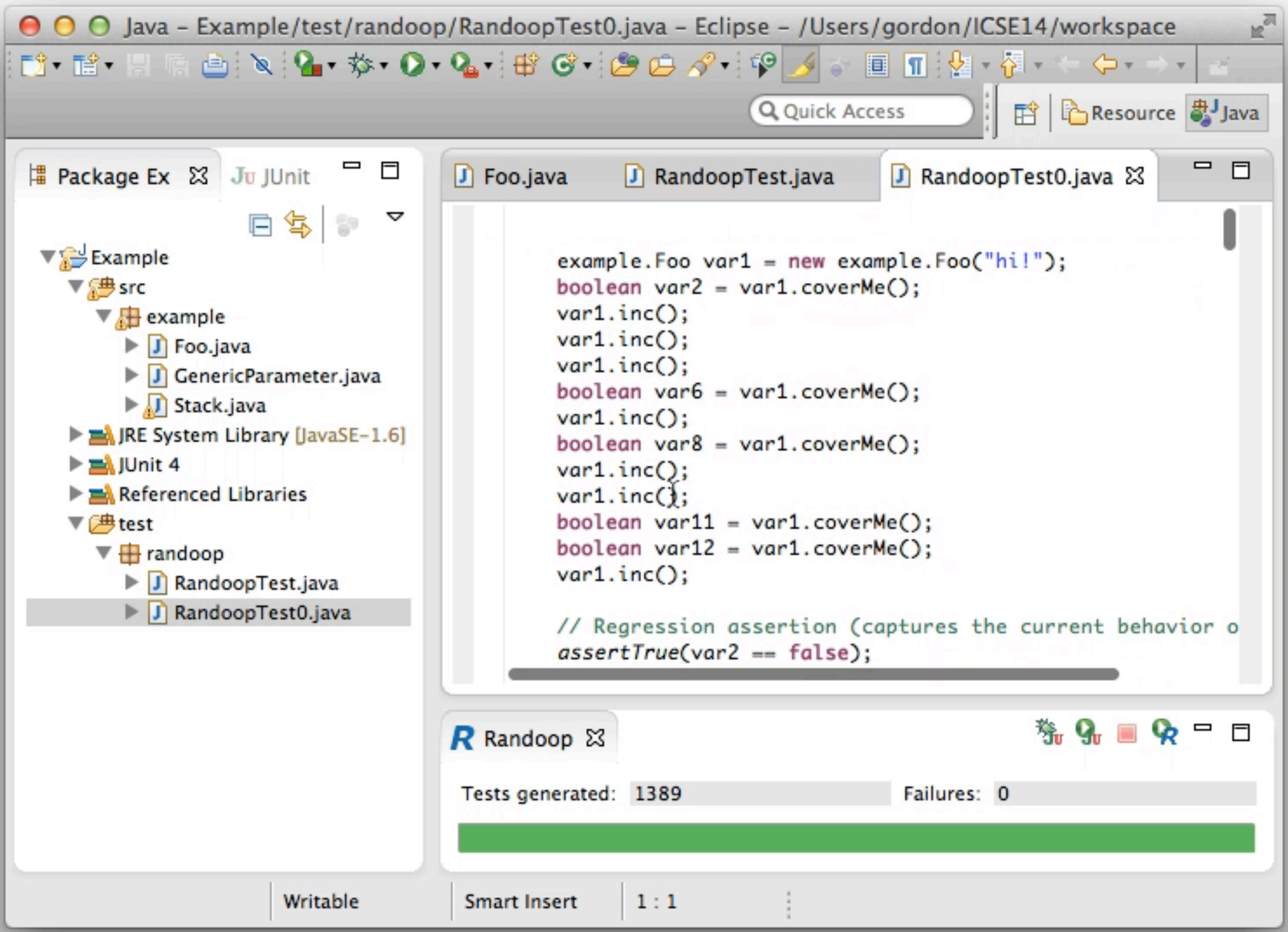
```java
package example;

public class Foo {
    private int x = 0;
    private String str;
    private String str2="bar";
    public Foo(String string) {
        this.str = string;
    }
    public void inc() {
        x++;
    }
    public boolean coverMe() {
        if (x==5)
            if(!str.equals(str2))
                if (str.equalsIgnoreCase(str2))
                    return true;

        return false;
    }
}
```

FooEvoSuiteTest.java

```java
 * This file was automatically generated by EvoS

package example;

import org.junit.Test;

public class FooEvoSuiteTest {


  @Test
  public void test0()  throws Throwable  {
      Foo foo0 = new Foo("bar");
      foo0.inc();
      foo0.inc();
      foo0.inc();
      foo0.inc();
      foo0.inc();
      boolean boolean0 = foo0.coverMe();
      assertEquals(false, boolean0);
  }

  @Test
  public void test1()  throws Throwable  {
```


Java – Example/src/example/Foo.java – Eclipse – /Users/gordon/ICSE14/workspace
Quick Access
Resource
Java
Package Ex
JUnit
Foo.java
Example
src
example
Foo.java
GenericParameter.java
Stack.java
JRE System Library [JavaSE-1.6]
JUnit 4
Referenced Libraries
example.Foo.java – Example/src

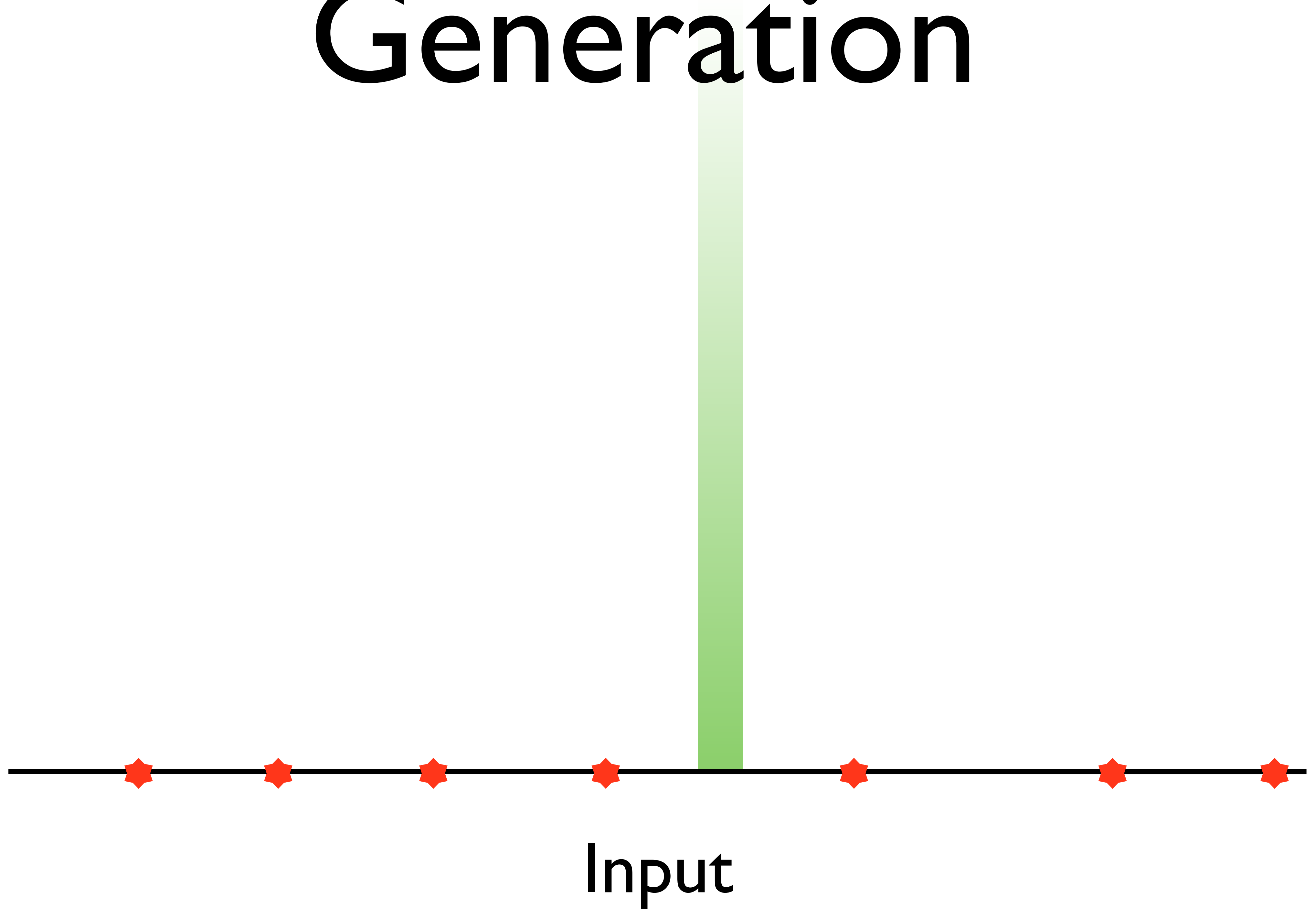
```
package example;

public class Foo {
    private int x = 0;
    private String str;
    private String str2="bar";
    public Foo(String string) {
        this.str = string;
    }
    public void inc() {
        x++;
    }
    public boolean coverMe() {
        if (x==5)
            if(!str.equals(str2))
                if (str.equalsIgnoreCase(str2))
                    return true;

        return false;
    }
}
```

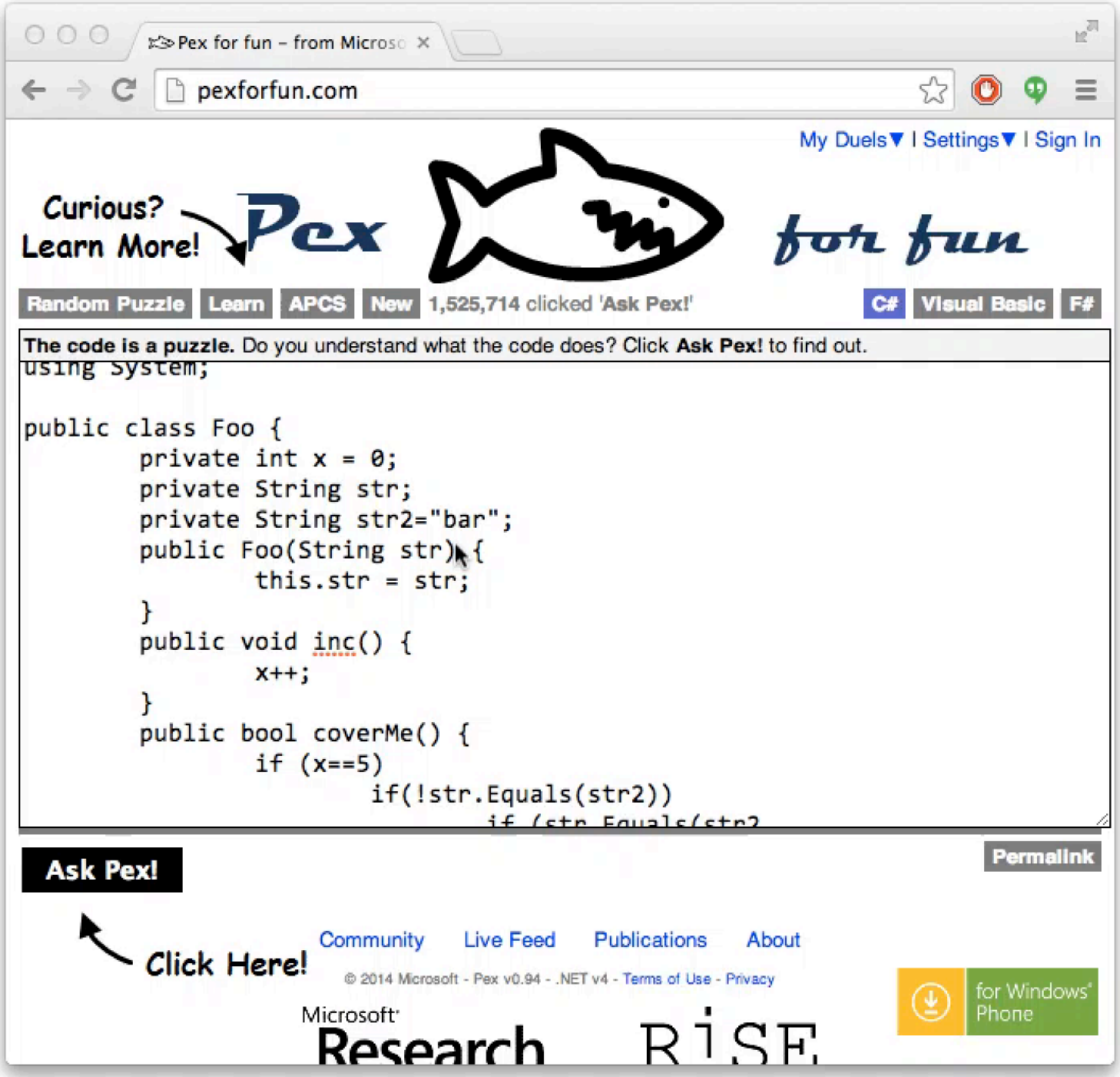


```
example.Foo var1 = new example.Foo("hi!");
boolean var2 = var1.coverMe();
var1.inc();
var1.inc();
var1.inc();
boolean var6 = var1.coverMe();
var1.inc();
boolean var8 = var1.coverMe();
var1.inc();
var1.inc();
boolean var11 = var1.coverMe();
boolean var12 = var1.coverMe();
var1.inc();

// Regression assertion (captures the current behavior o
assertTrue(var2 == false);
```

Tests generated: 1389 Failures: 0

Random Test Data Generation



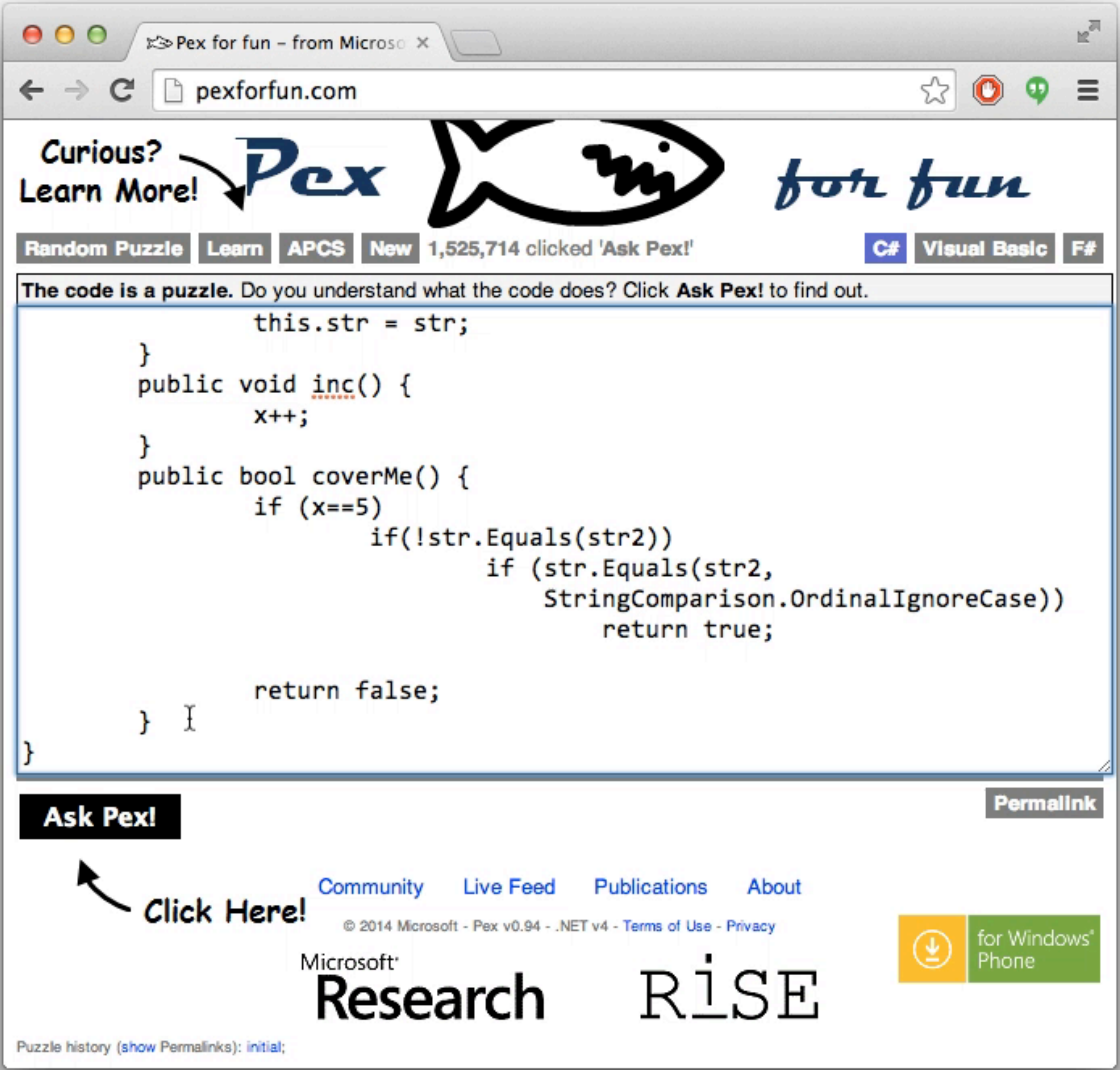


1,525,714 clicked 'Ask Pex!'

The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
using System;

public class Foo {
        private int x = 0;
        private String str;
        private String str2="bar";
        public Foo(String str) {
                this.str = str;
        }
        public void inc() {
                x++;
        }
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2
```

© 2014 Microsoft - Pex v0.94 - .NET v4 - Terms of Use - Privacy



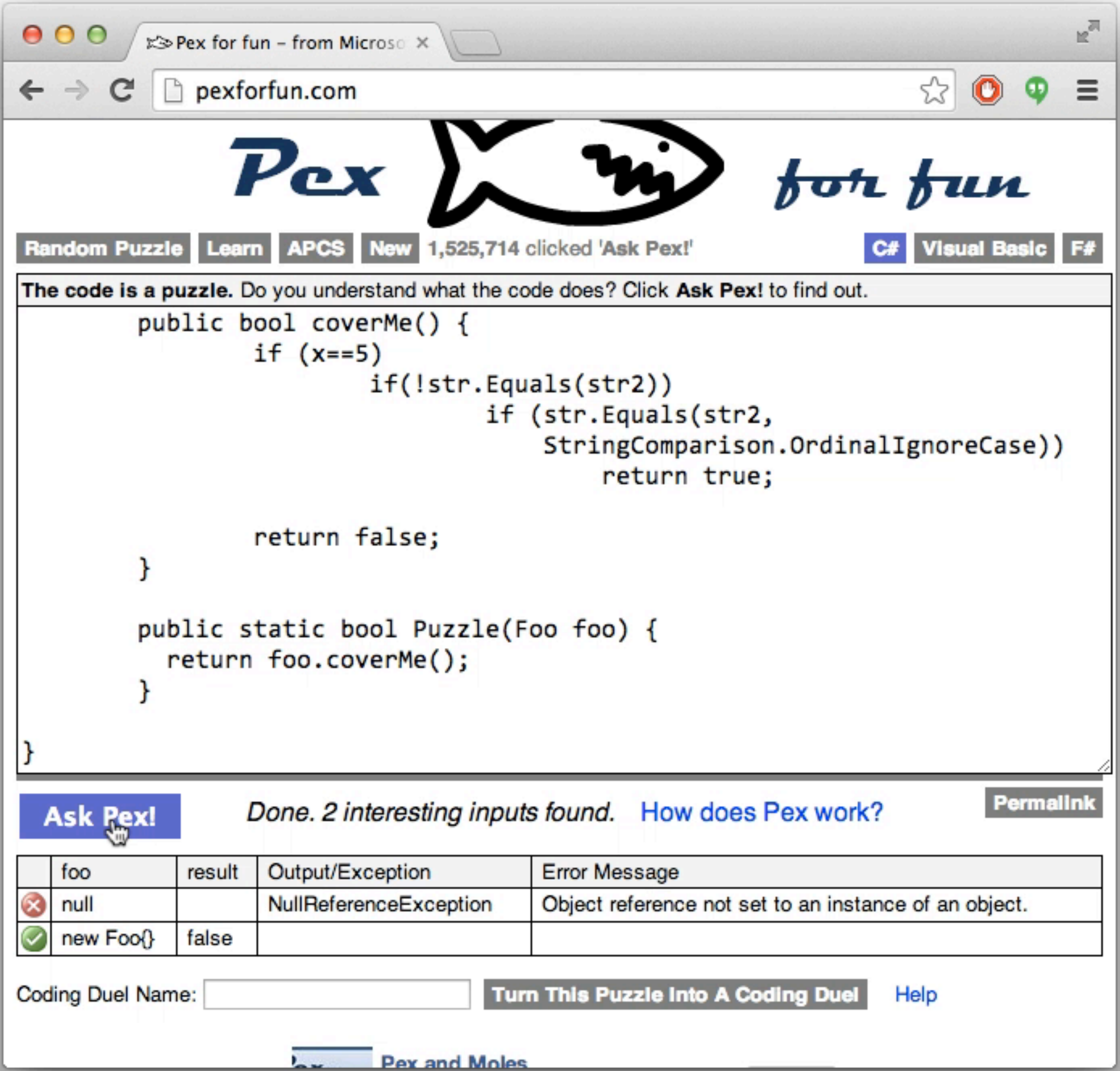
1,525,714 clicked 'Ask Pex!'

The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
                this.str = str;
        }
        public void inc() {
                x++;
        }
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2,
                                    StringComparison.OrdinalIgnoreCase))
                                        return true;

                return false;
        }
}
```

© 2014 Microsoft - Pex v0.94 - .NET v4 - Terms of Use - Privacy

Puzzle history (show Permalinks): initial;



The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2,
                                    StringComparison.OrdinalIgnoreCase))
                                        return true;

                return false;
        }

        public static bool Puzzle(Foo foo) {
          return foo.coverMe();
        }

}
```

Done. 2 interesting inputs found. How does Pex work?

	foo	result	Output/Exception	Error Message
✗	null		NullReferenceException	Object reference not set to an instance of an object.
✓	new Foo()	false		

Coding Duel Name: Turn This Puzzle Into A Coding Duel

Generating vs Checking

Conventional Software Testing Research

Write a method to construct test cases

Search-Based Testing

Write a method
to determine how good a test case is

Generating vs Checking

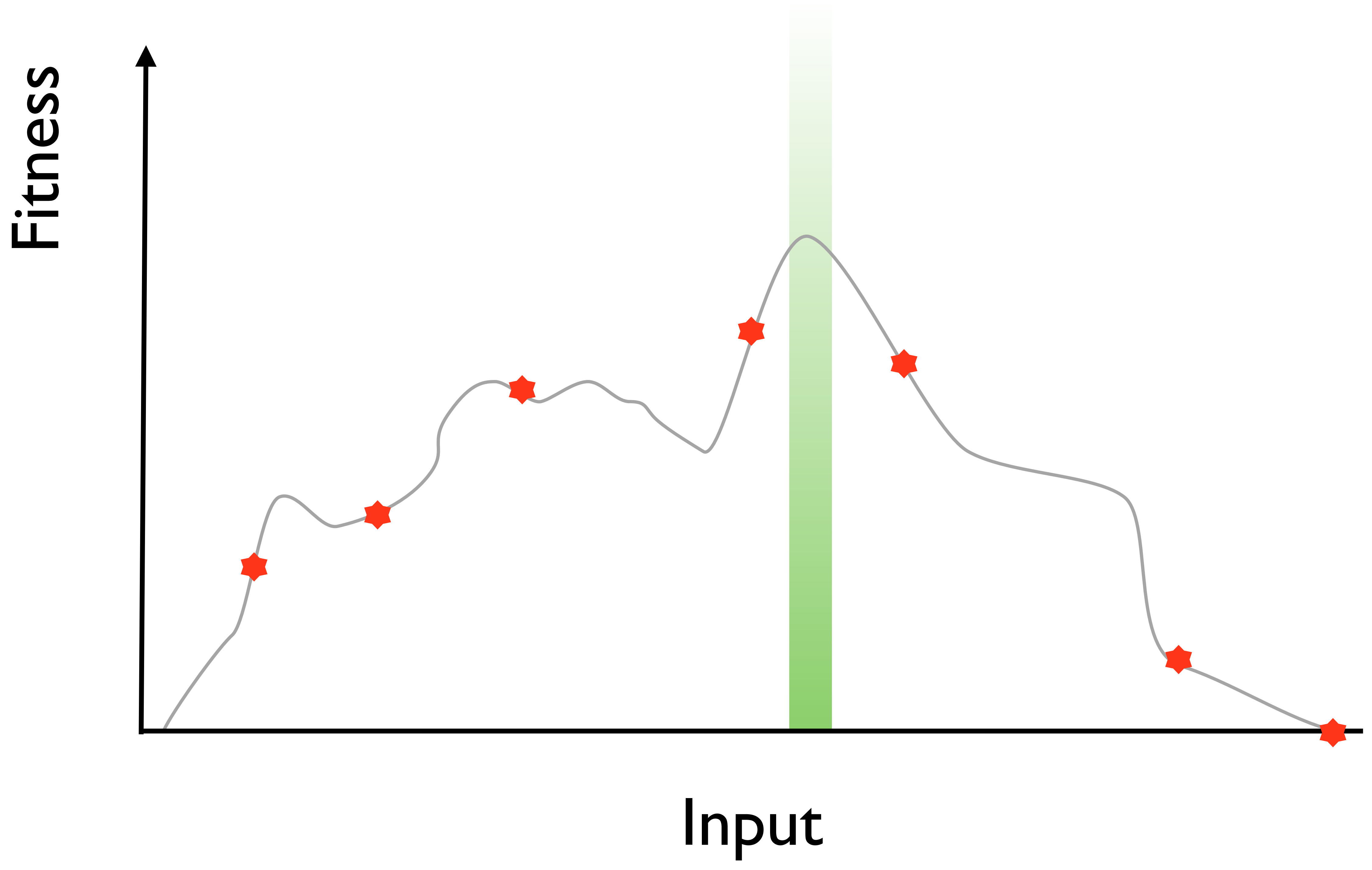
Conventional Software Testing Research

Write a method to construct test cases

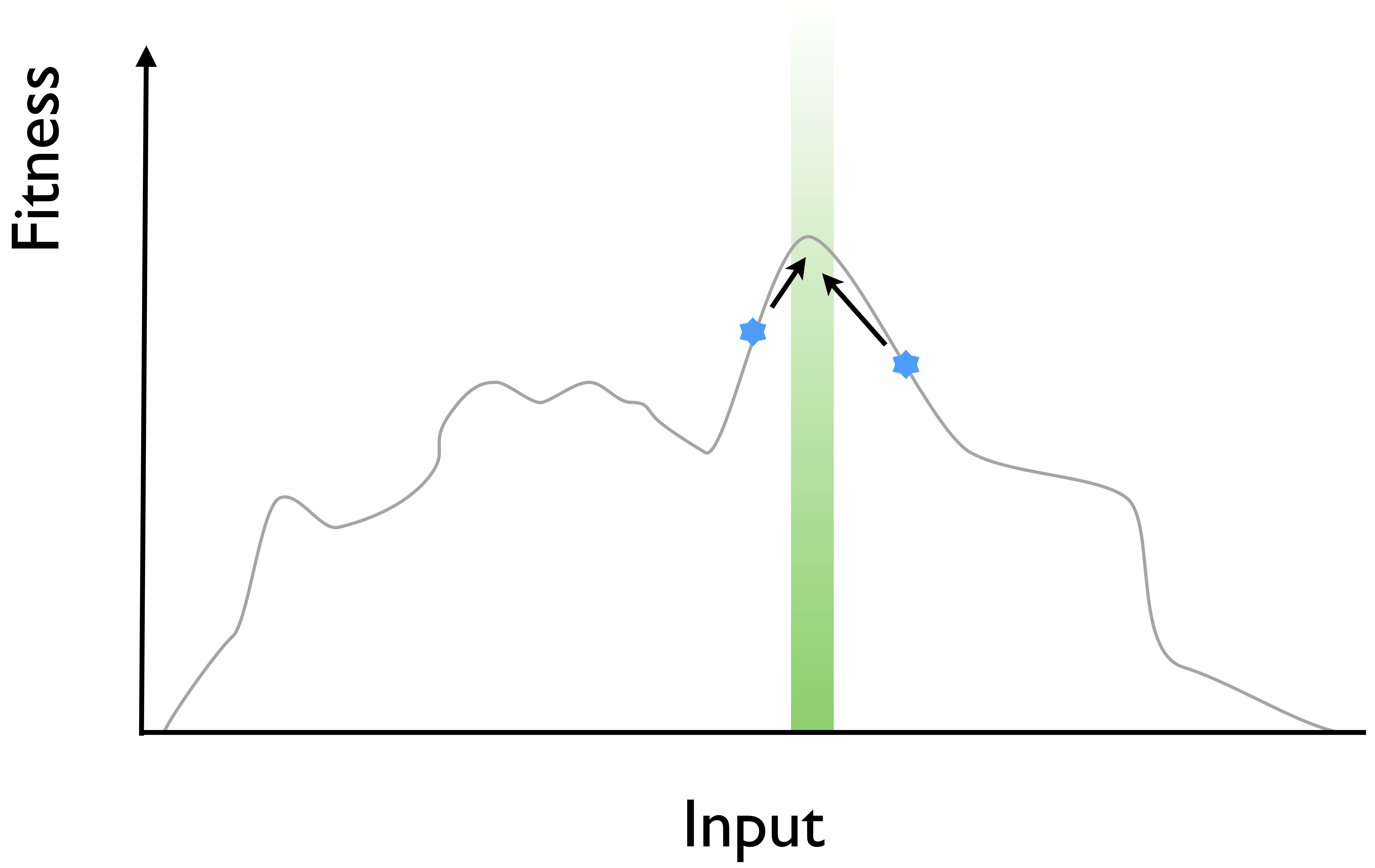
Search-Based Testing

Write a fitness function
to determine how good a test case is

Fitness-guided search

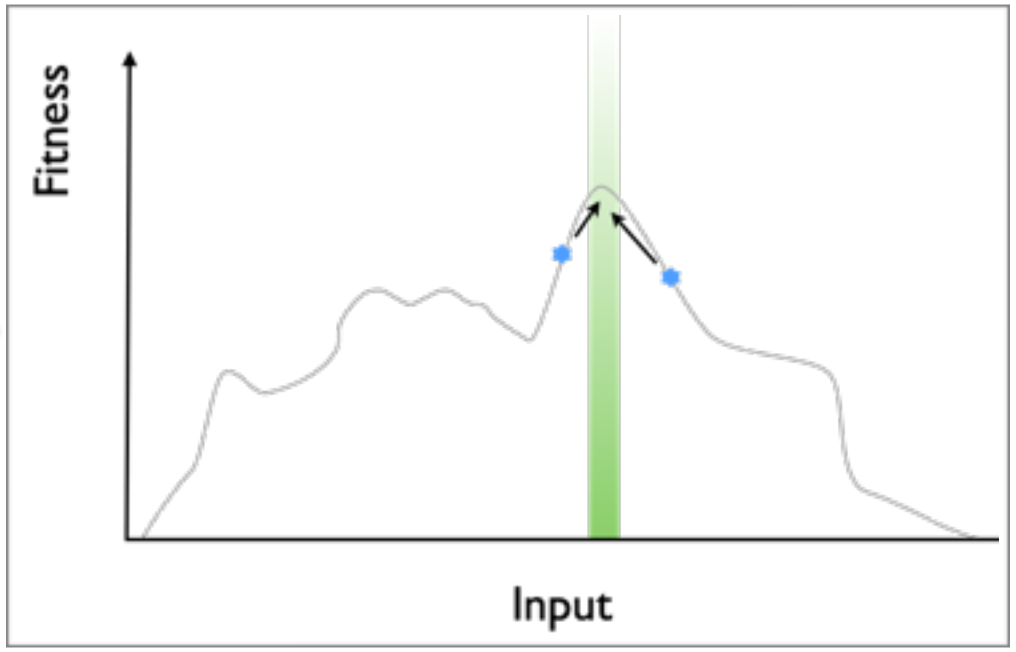


Fitness-guided search



Components of an SBST Tool

```
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```



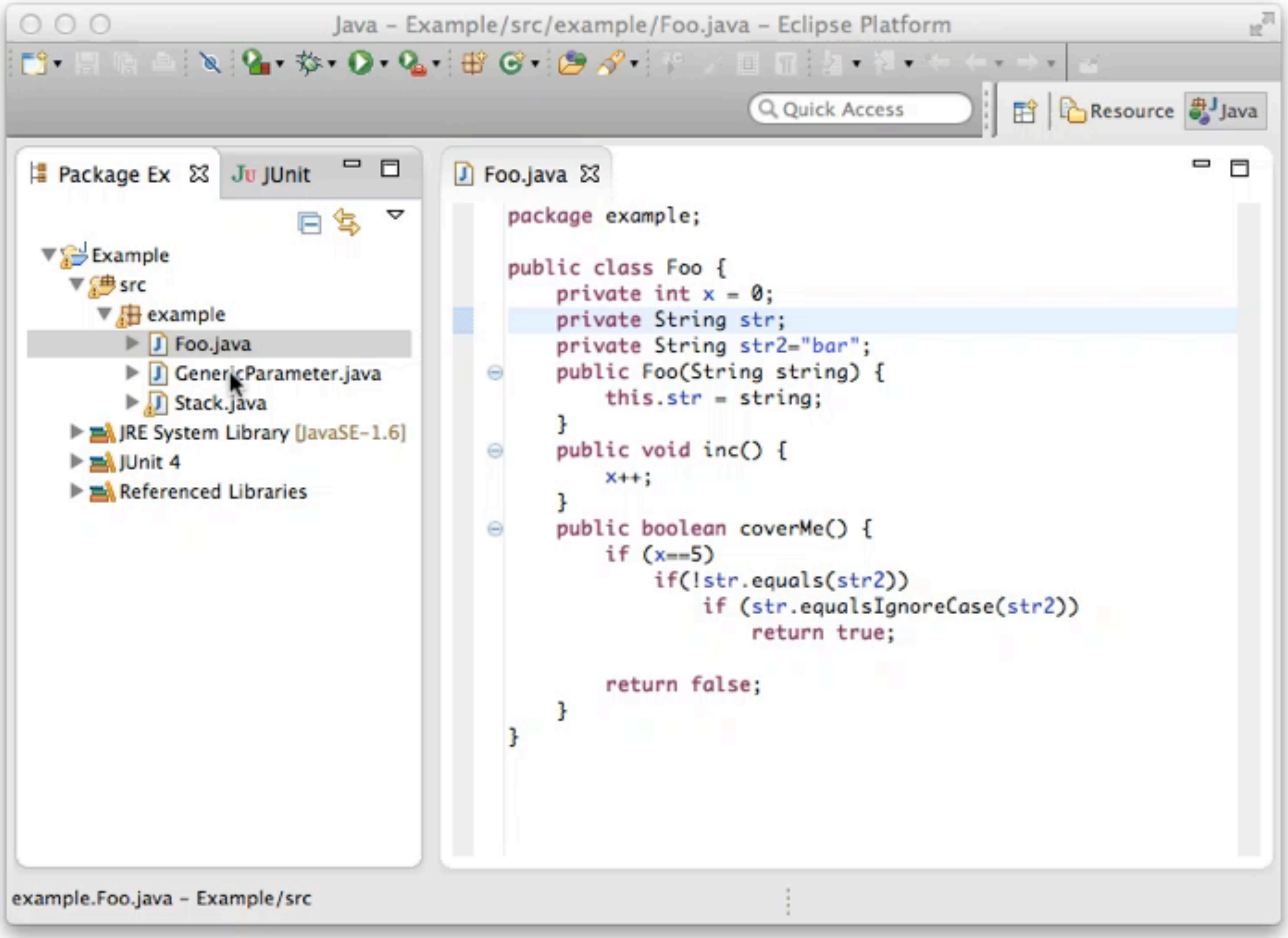
Search Algorithm

Representation

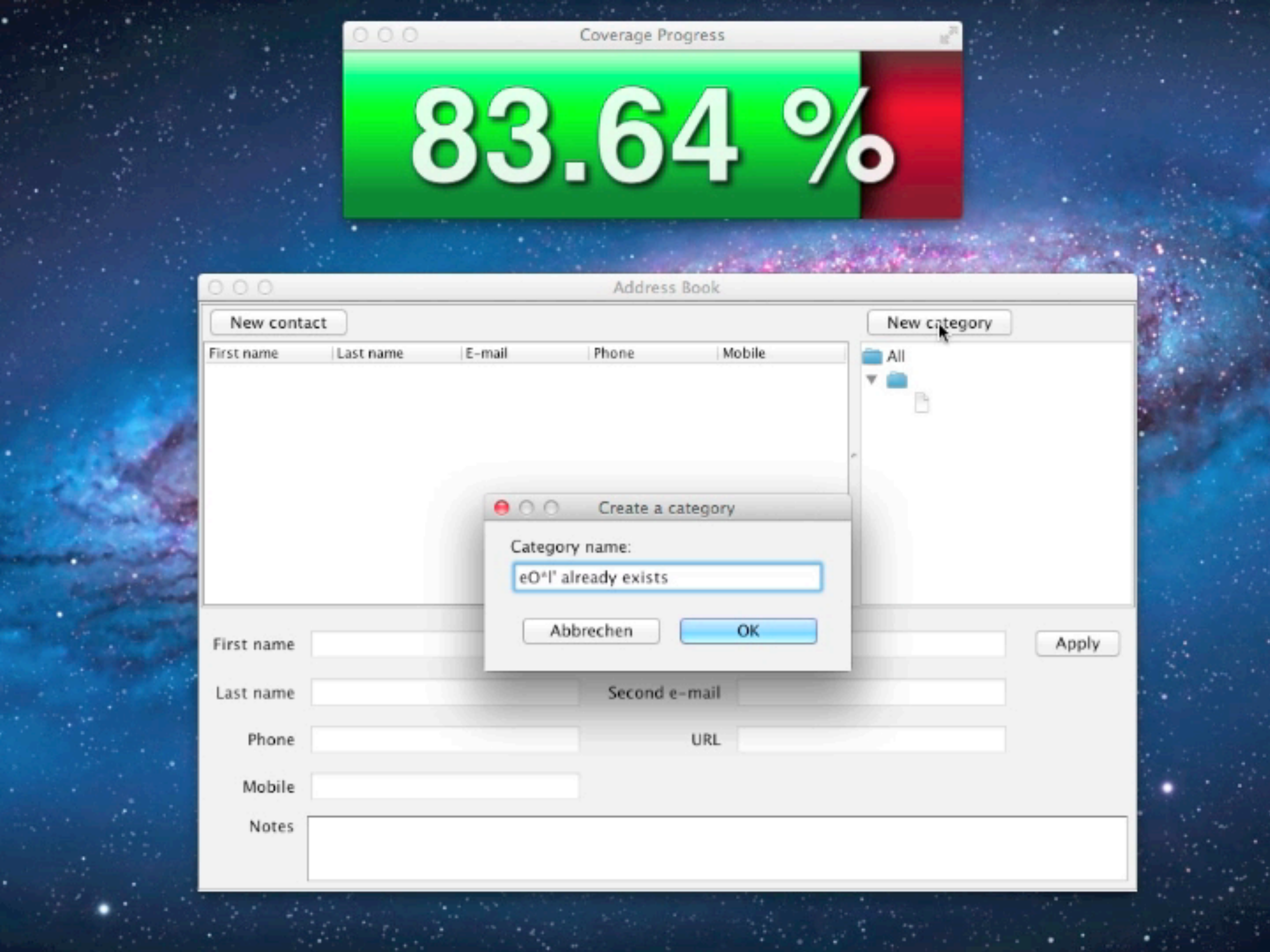
Fitness Function

Components of an SBST Tool

Search Algorithm	Meta-heuristic algorithm
Representation	Encoding of the problem solution
Search Operators	Modifications of encoded solutions
Fitness Function	Measure how good a candidate solution is
Test Execution	Execute tests
Instrumentation	Collect data/traces for fitness calculation during execution



```java
package example;

public class Foo {
    private int x = 0;
    private String str;
    private String str2="bar";
    public Foo(String string) {
        this.str = string;
    }
    public void inc() {
        x++;
    }
    public boolean coverMe() {
        if (x==5)
            if(!str.equals(str2))
                if (str.equalsIgnoreCase(str2))
                    return true;

        return false;
    }
}
```



Coverage Progress

83.64 %

Address Book

Create a category

Category name:

eO*l' already exists

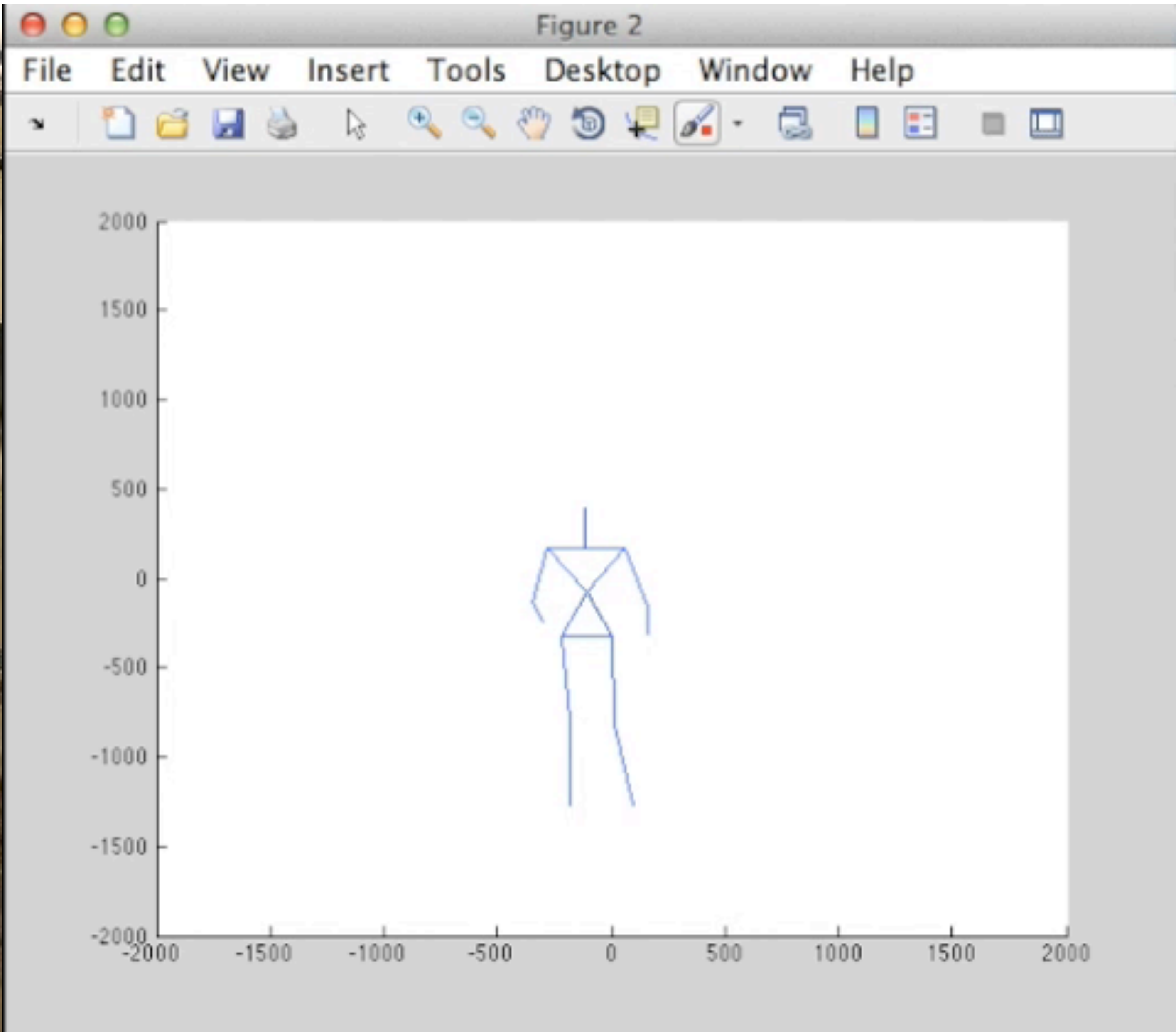


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2000
-1500
-1000
-500
0
500
1000
1500
2000

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

Outline

```python
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```

Components of an SBST Tool

Search Algorithm

Hill-climbing

Representation

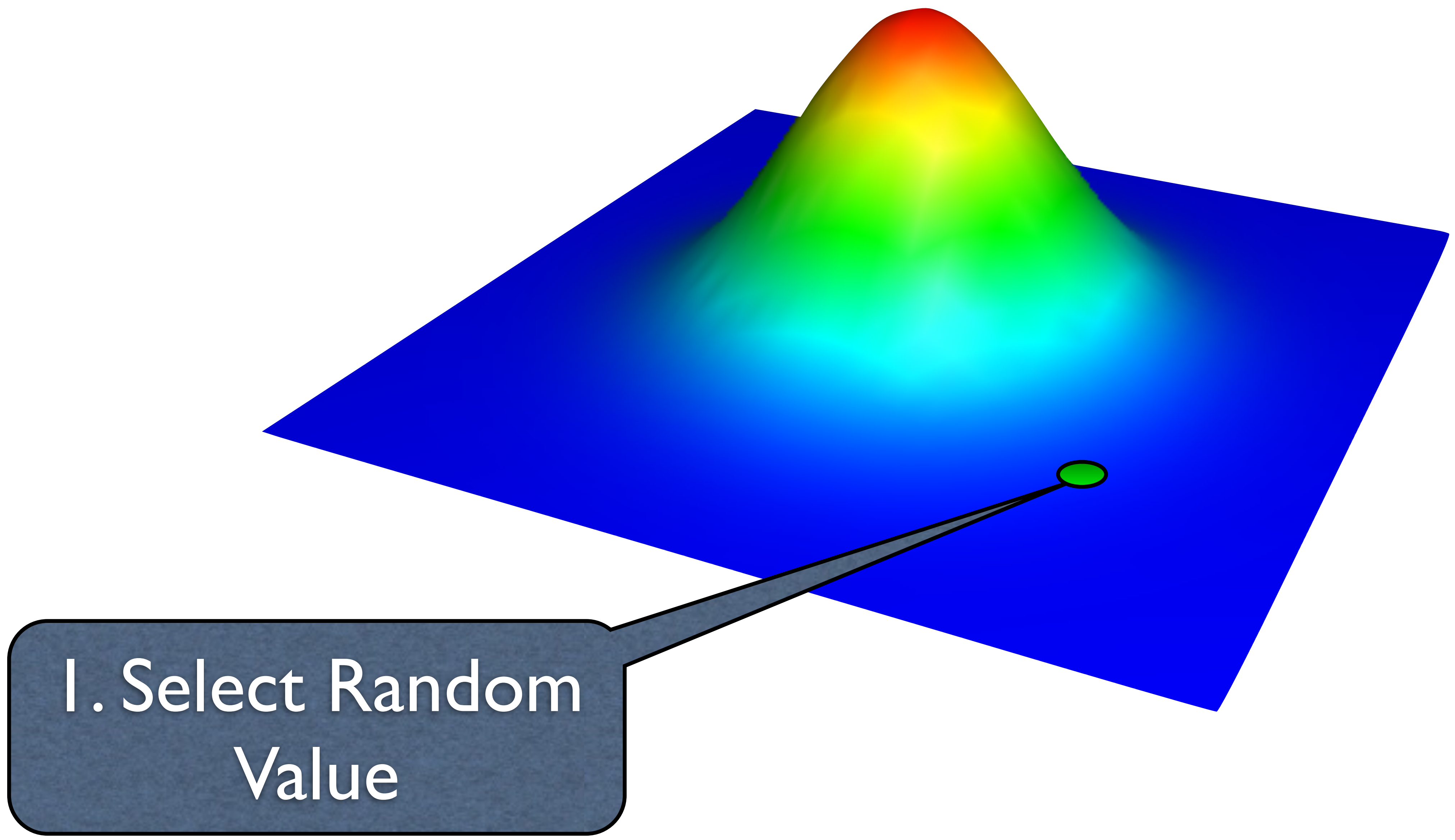
Search Operators

Fitness Function

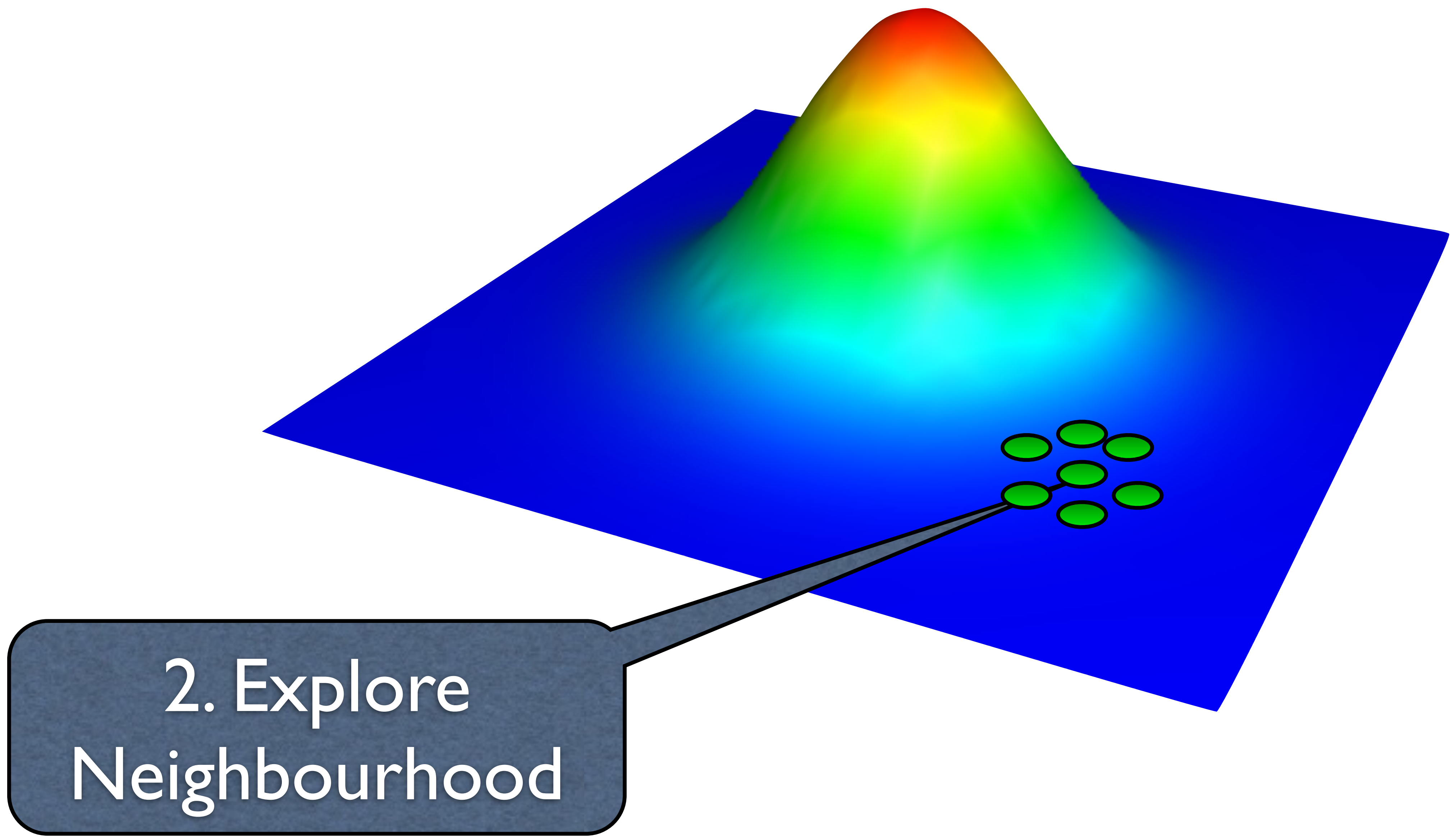
Test Execution

Instrumentation

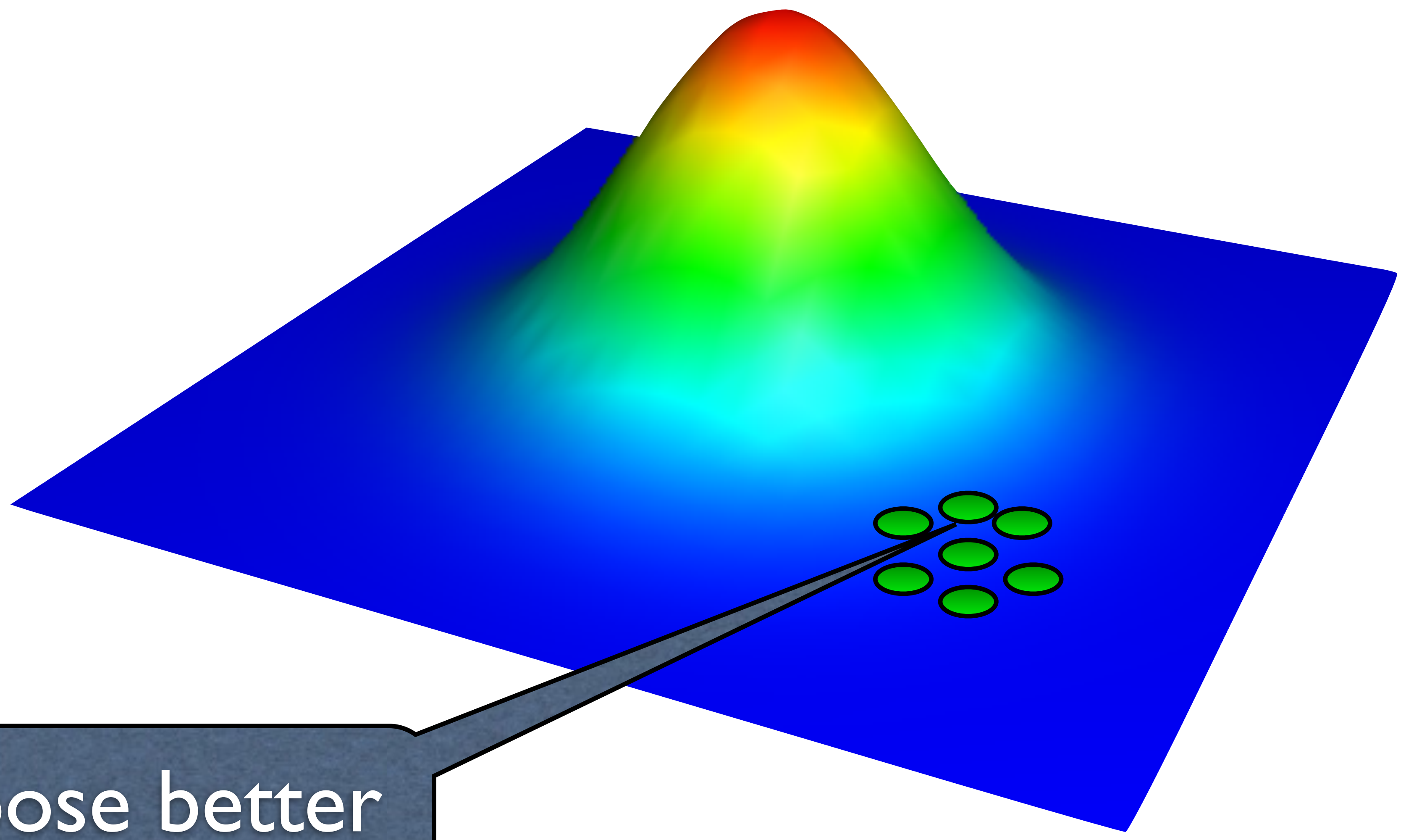
Hill Climbing



Hill Climbing

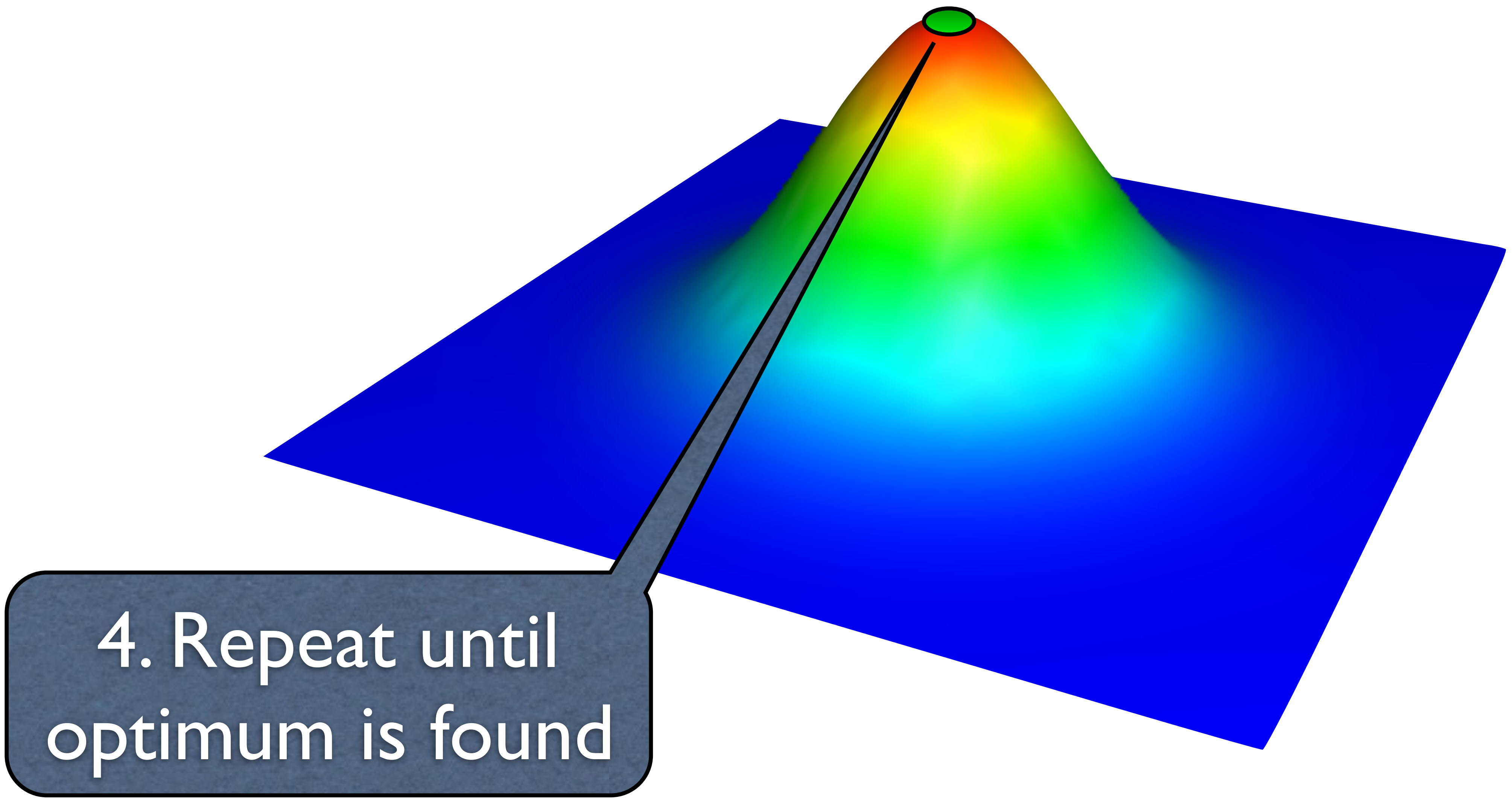


Hill Climbing



3. Choose better neighbour

Hill Climbing

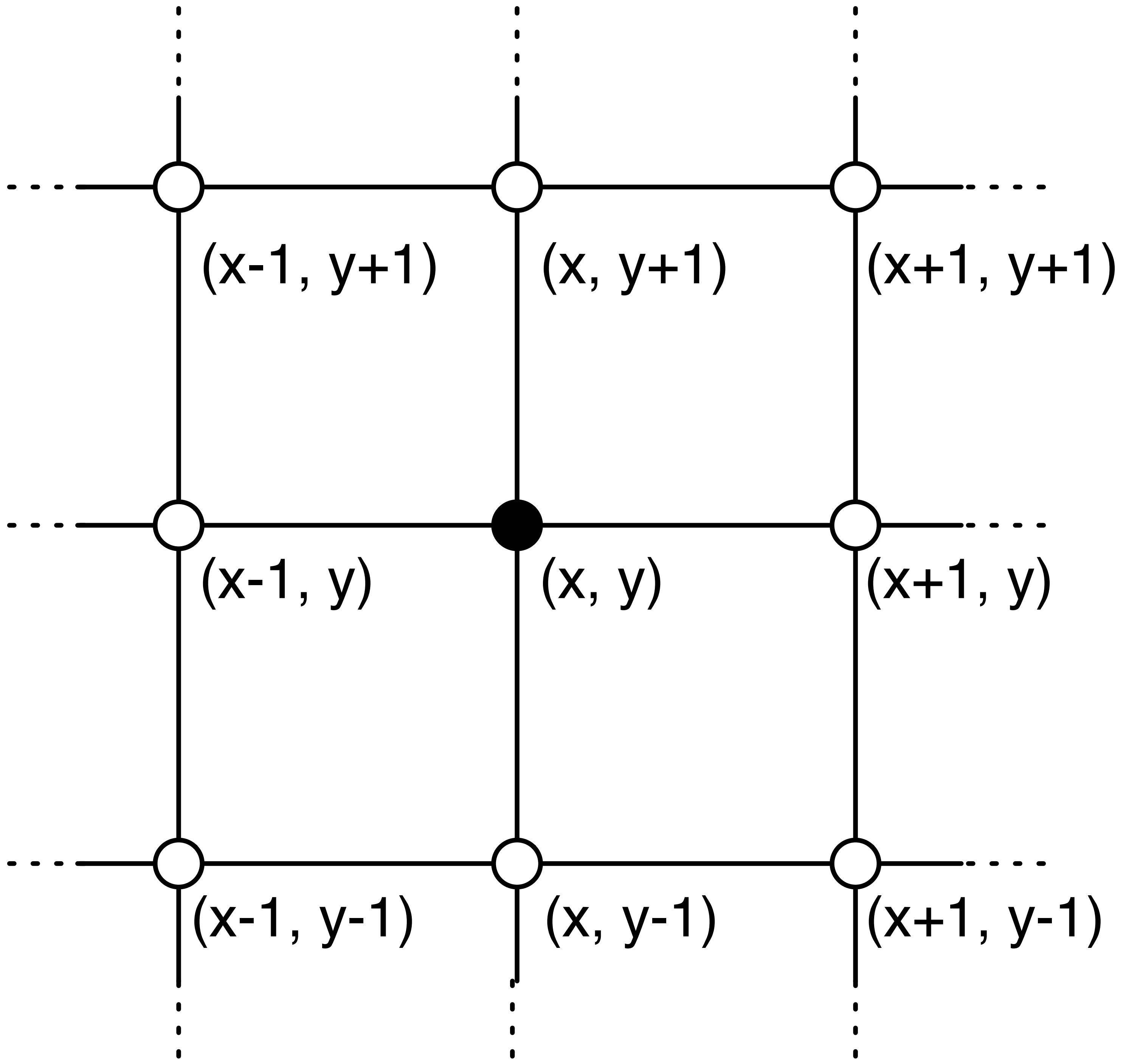


Components of an SBST Tool

- Search Algorithm
- Representation
- Search Operators
- Fitness Function
- Test Execution
- Instrumentation

Hill-climbing

```
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```

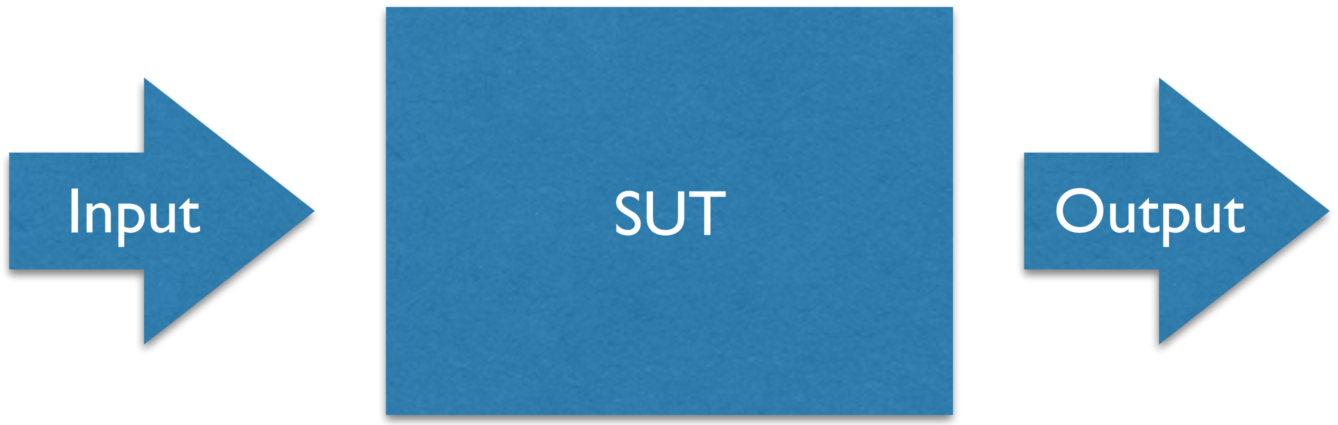


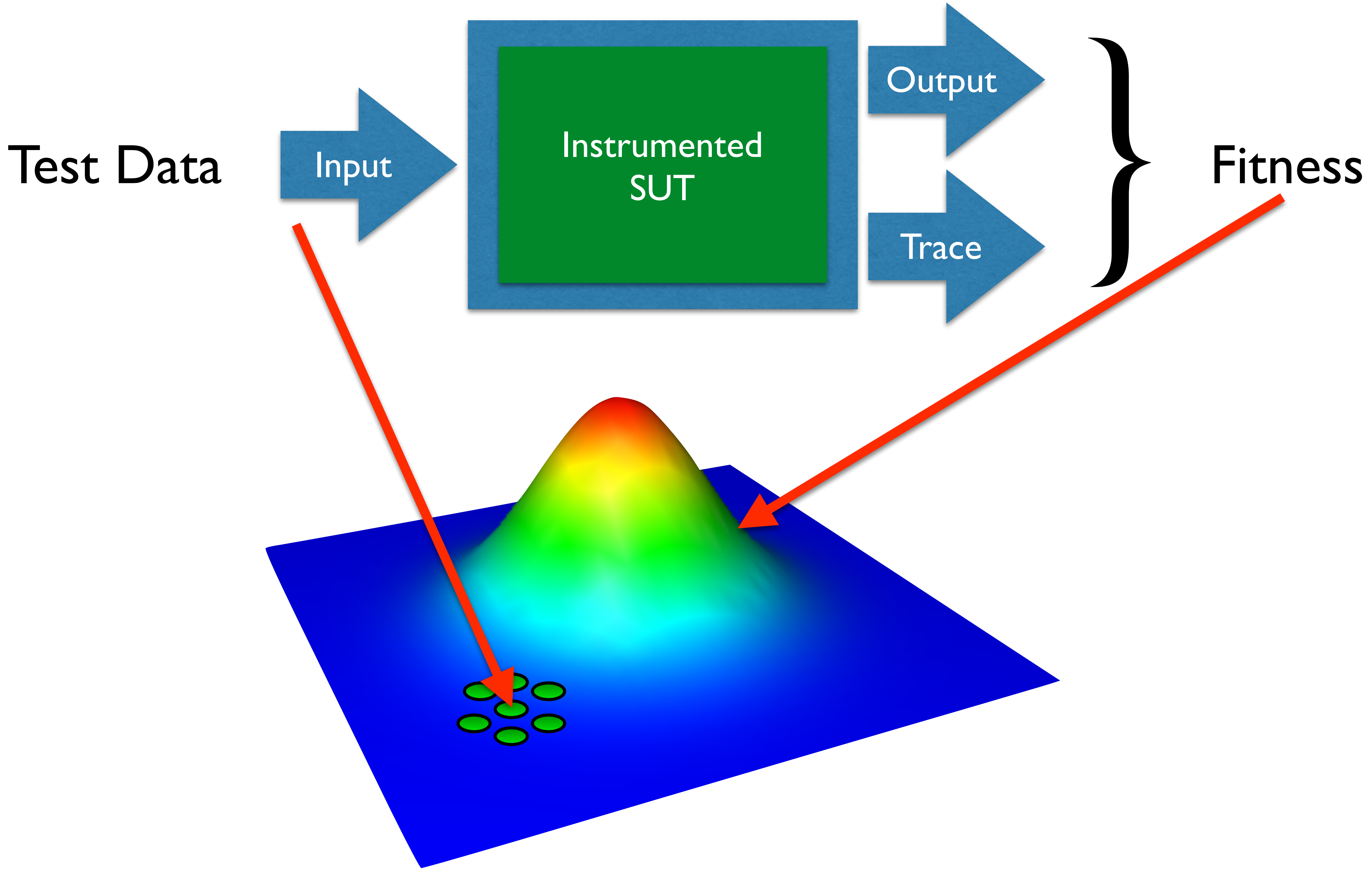
Components of an SBST Tool

Search Algorithm	Hill-climbing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	
Test Execution	
Instrumentation	

Components of an SBST Tool

Search Algorithm	Hill-climbing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	
Test Execution	
Instrumentation	

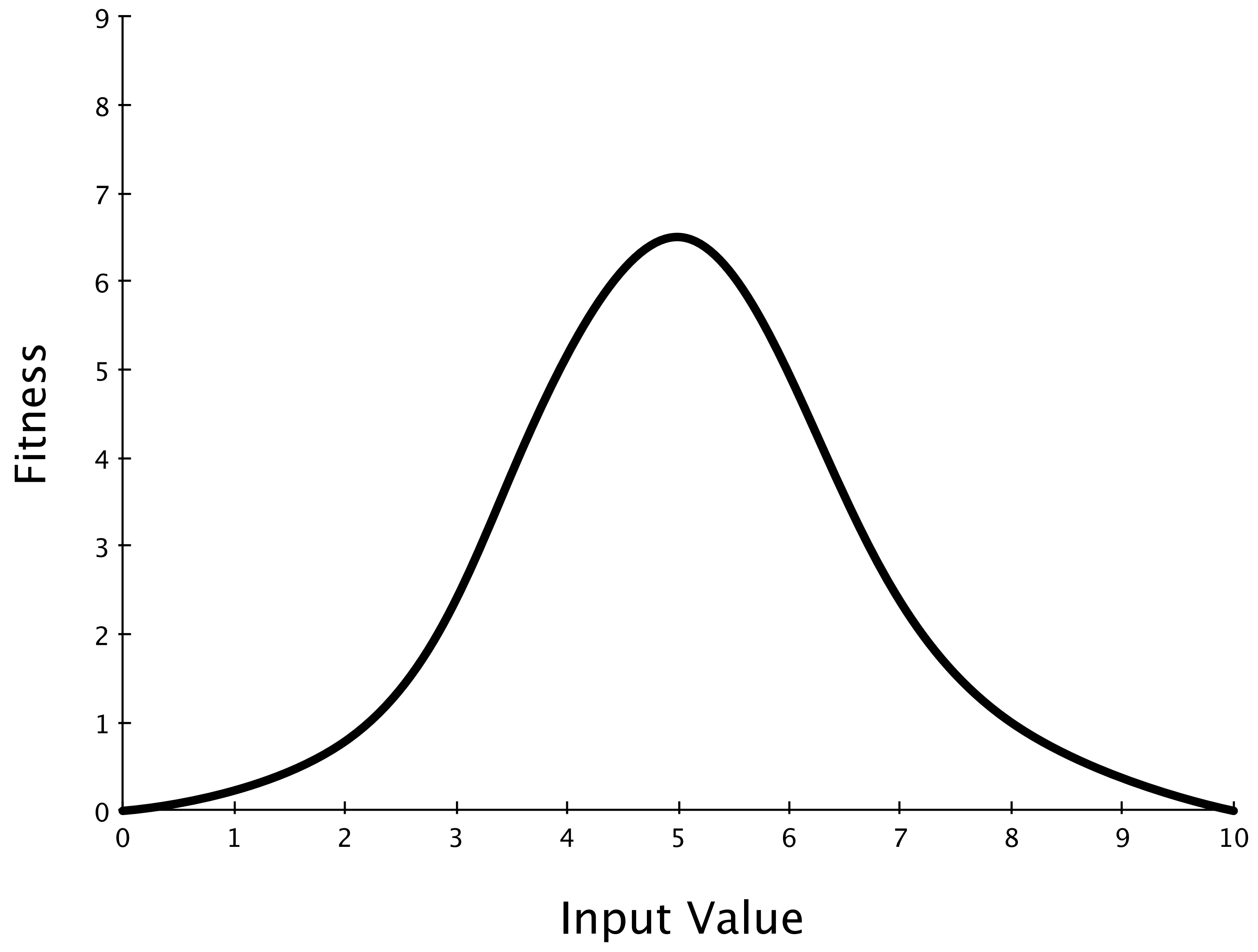

Input
SUT
Output


Test Data
Input
Instrumented SUT
Output
Trace
Fitness

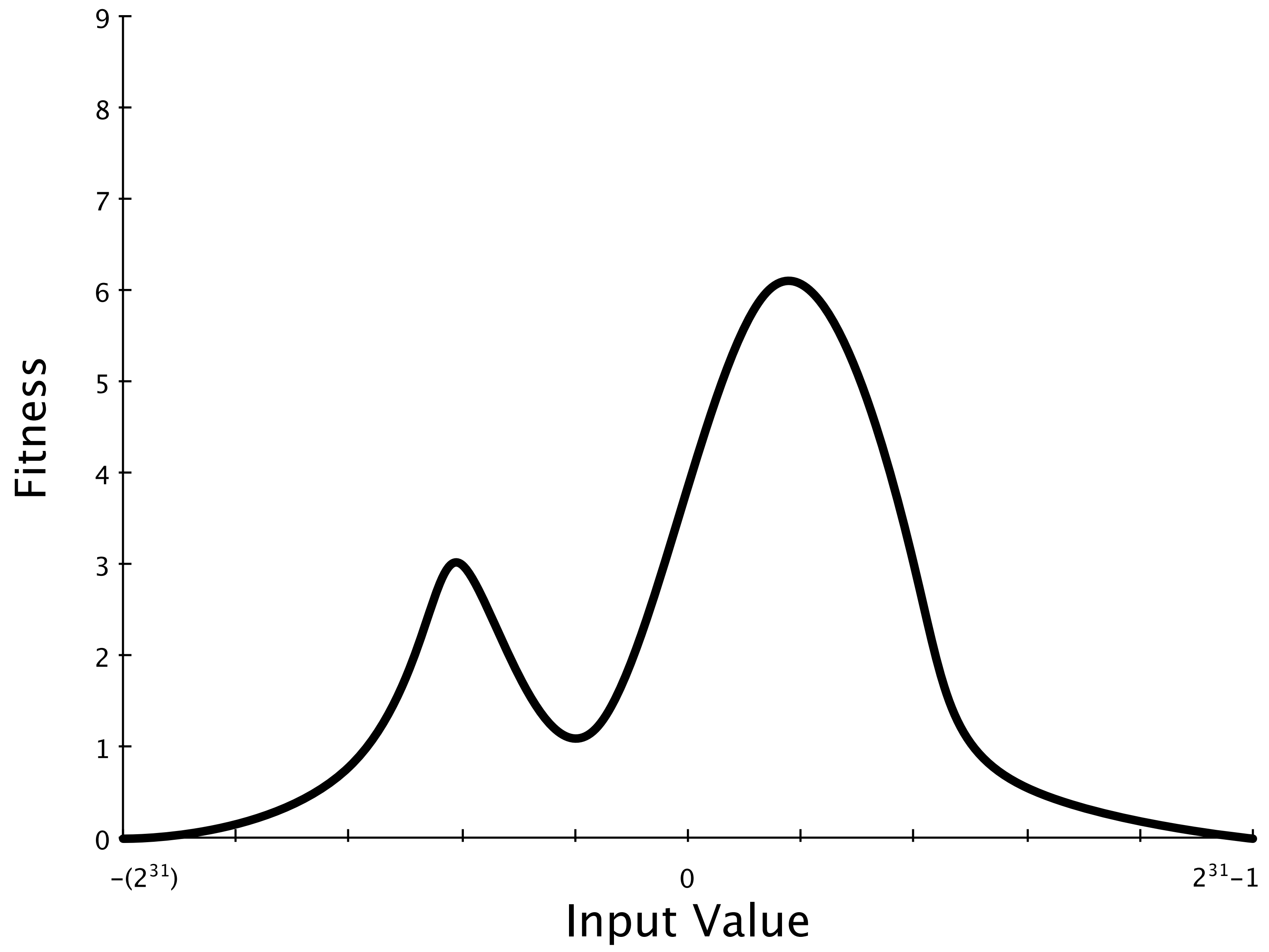
```python
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```

Components of an SBST Tool

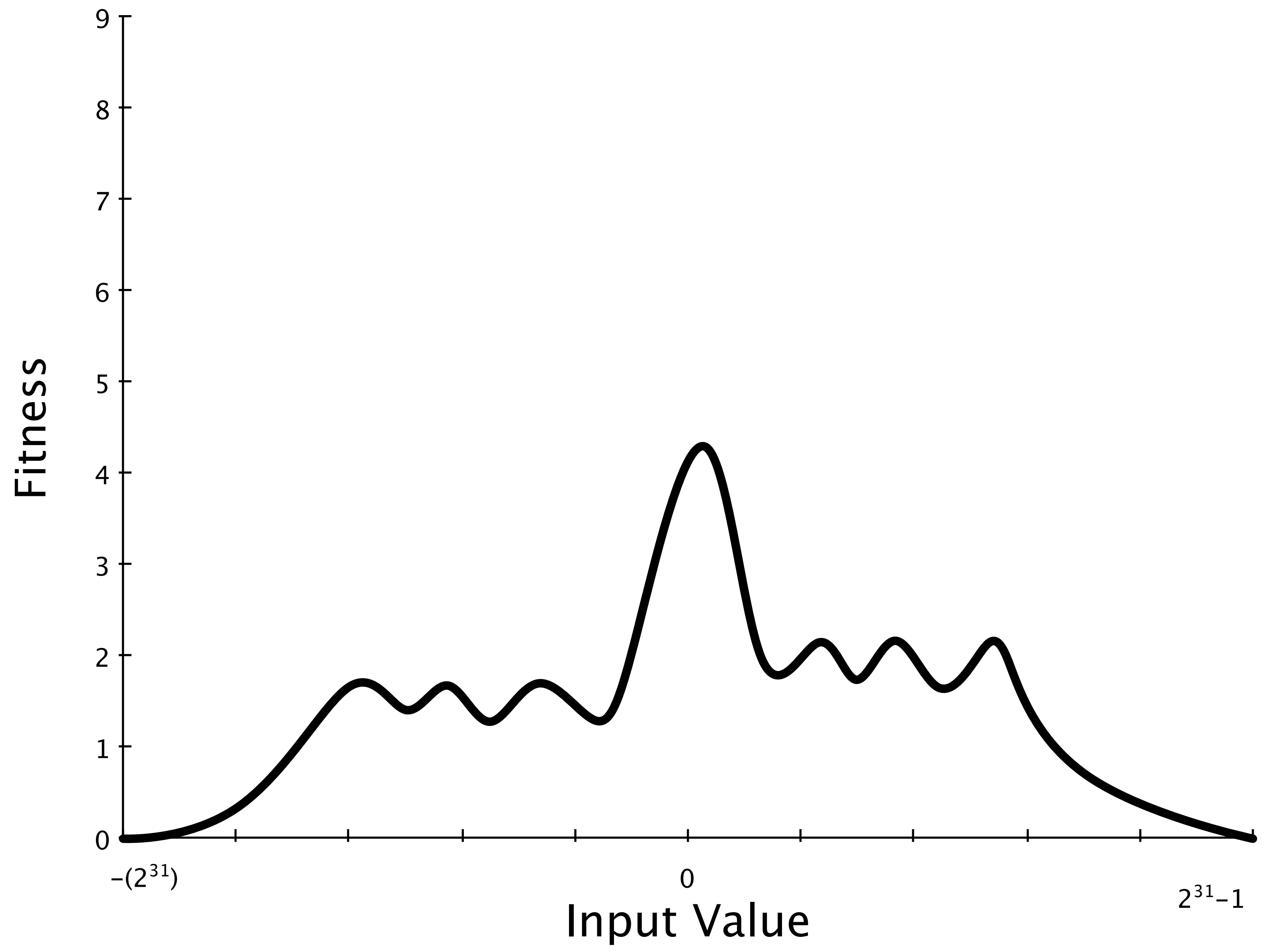
Search Algorithm	Hill-climbing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	Global variable

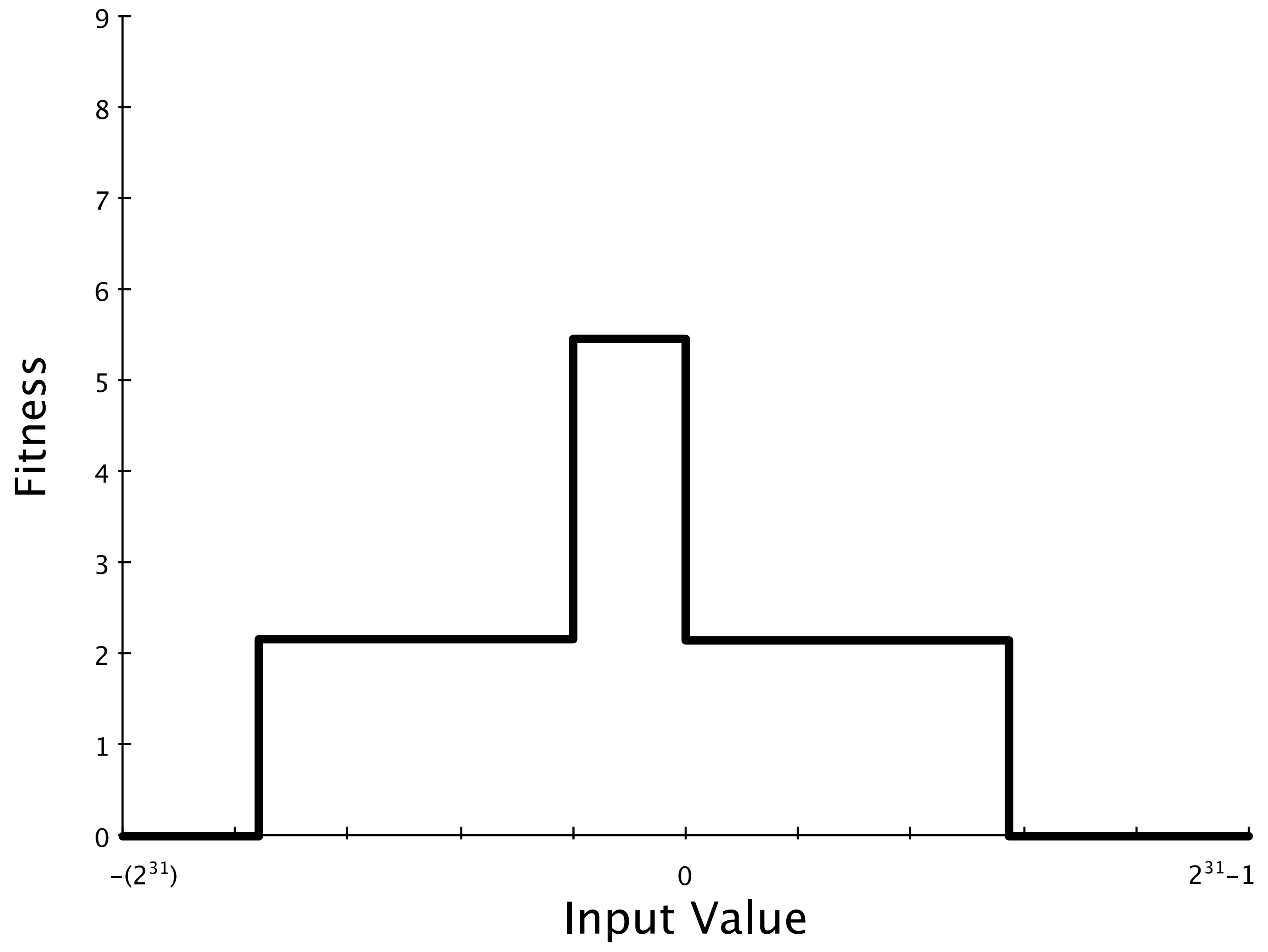


Fitness
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
Input Value



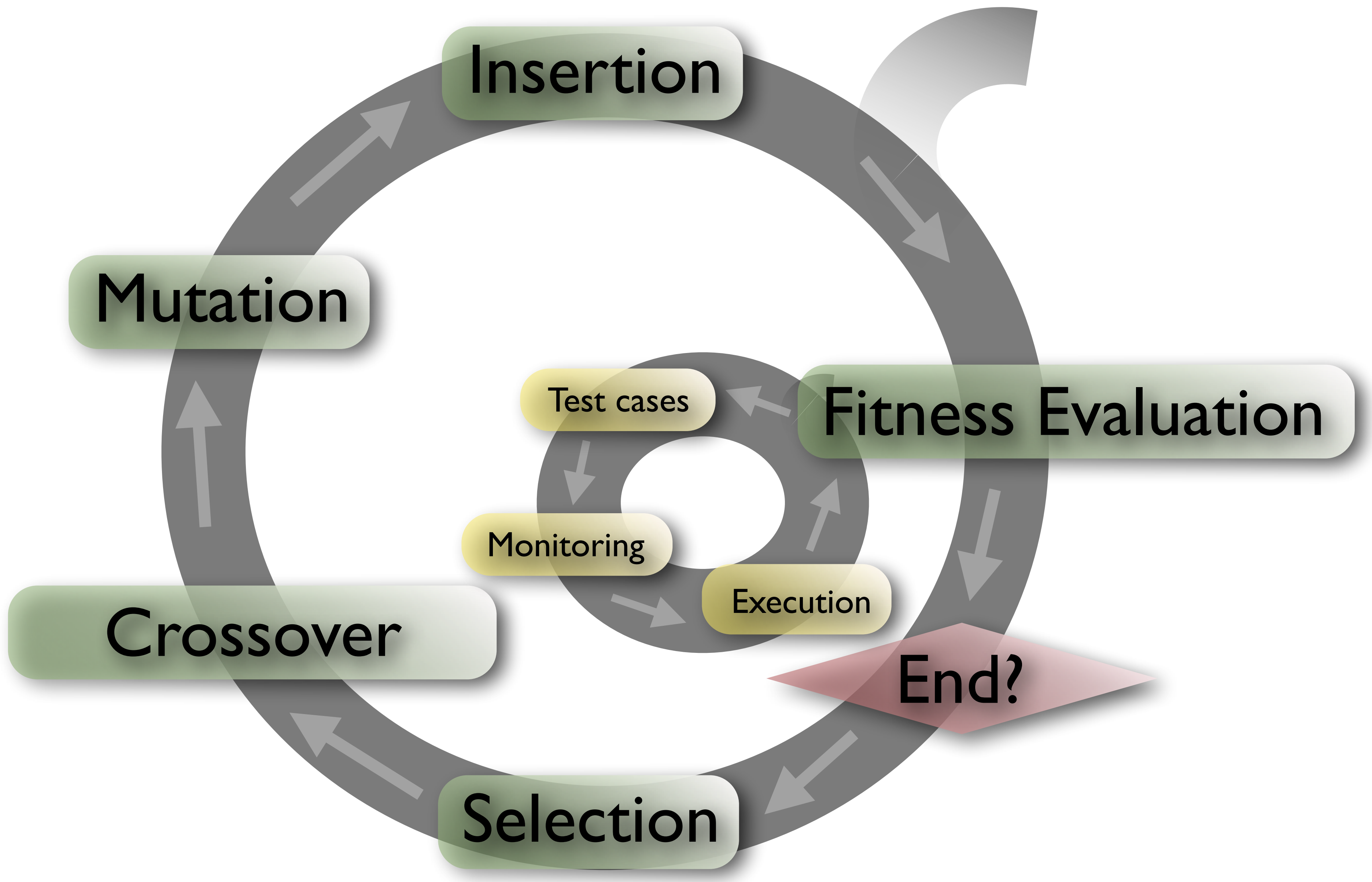
9
8
7
6
5
4
3
2
1
0
Fitness
−(2³¹)
0
2³¹−1
Input Value

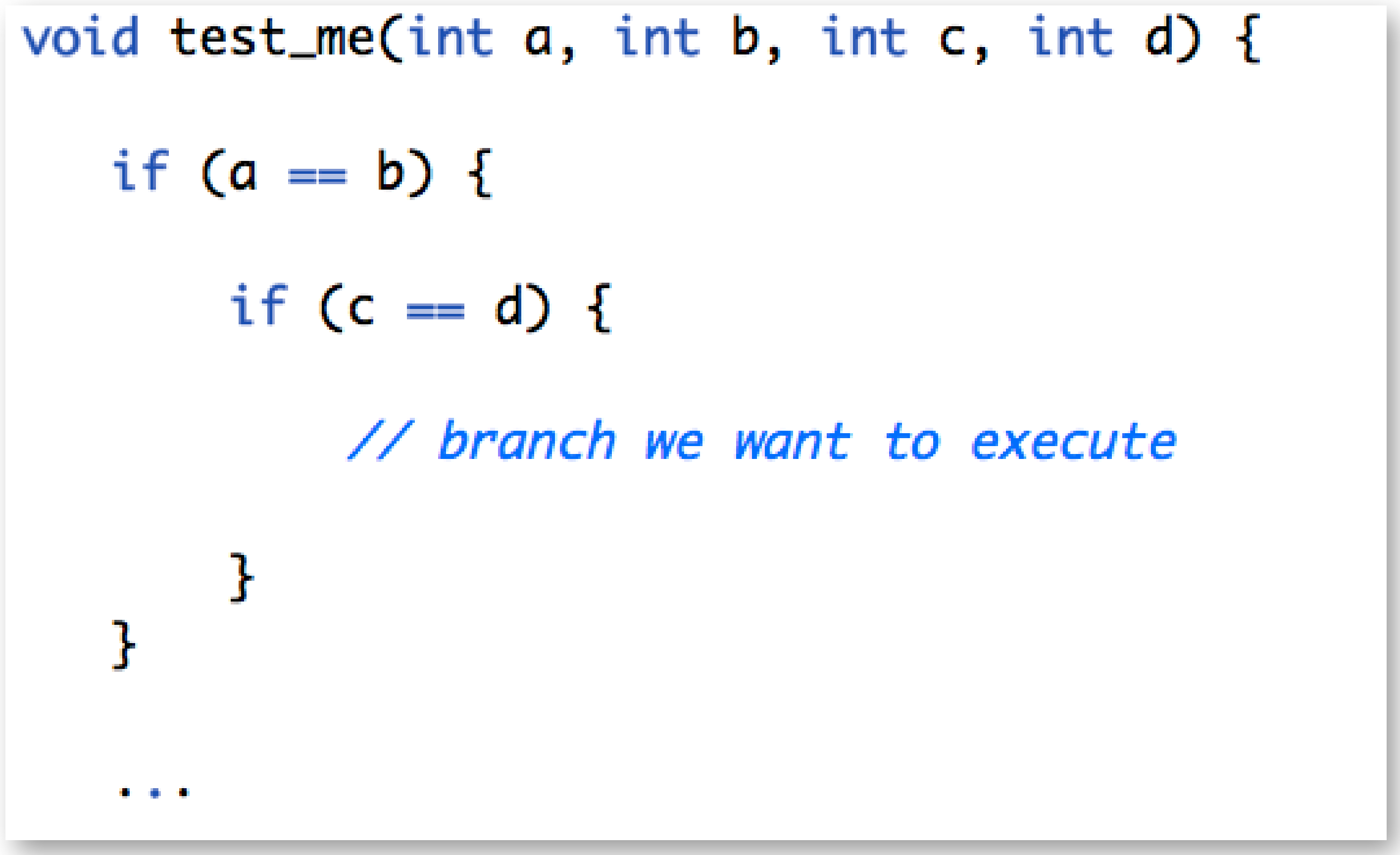

Fitness
9
8
7
6
5
4
3
2
1
0
−(2^31)
0
2^31−1
Input Value



Fitness
9
8
7
6
5
4
3
2
1
0
−(2^31)
0
2^31−1
Input Value

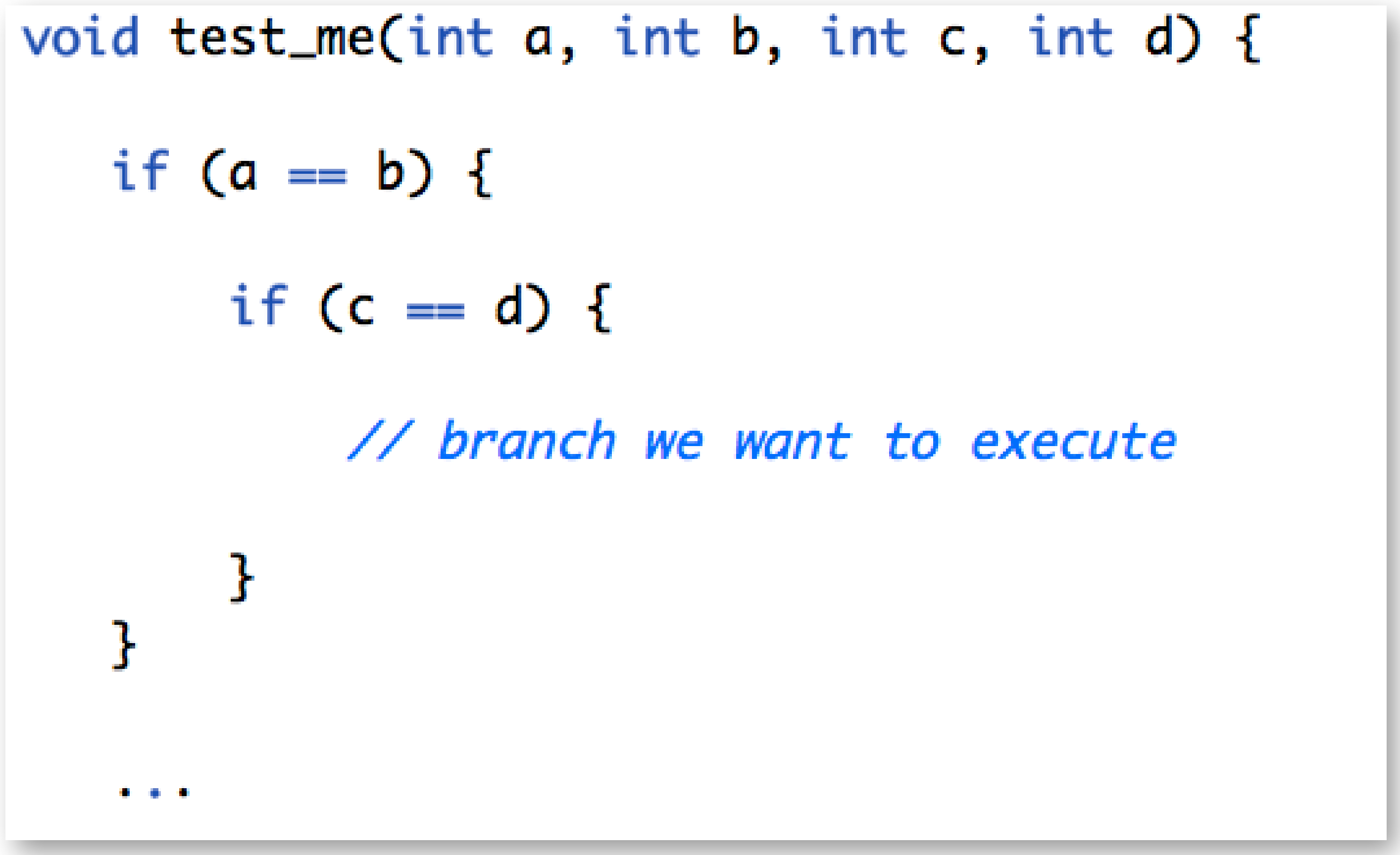
Evolutionary Testing



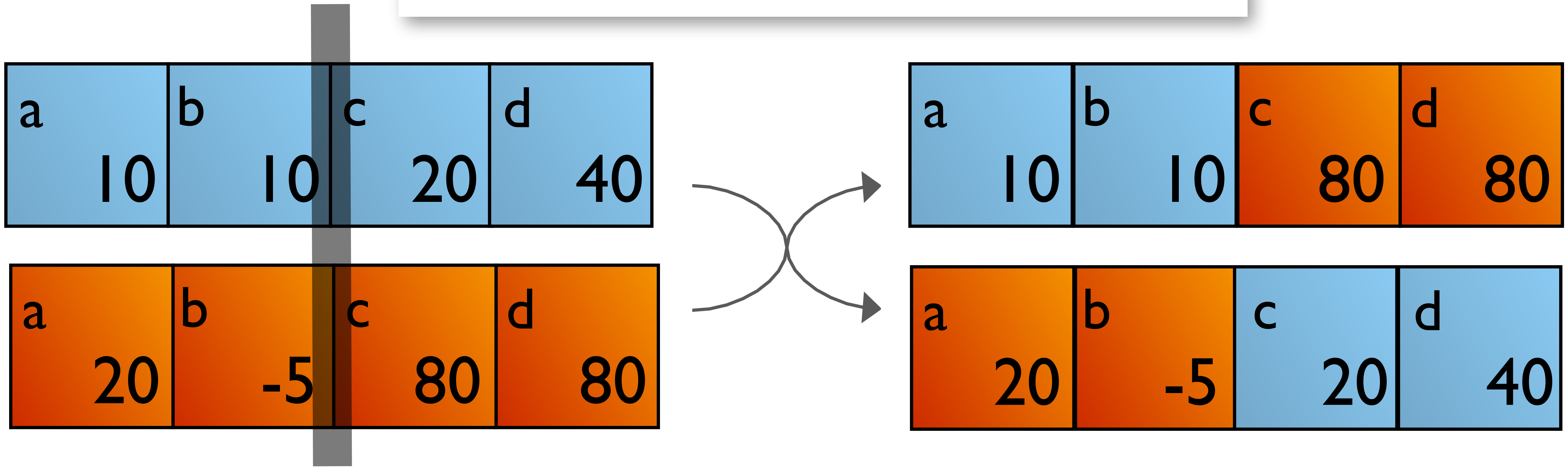


```c
void test_me(int a, int b, int c, int d) {

   if (a == b) {

       if (c == d) {

           // branch we want to execute

       }
   }

   ...
```

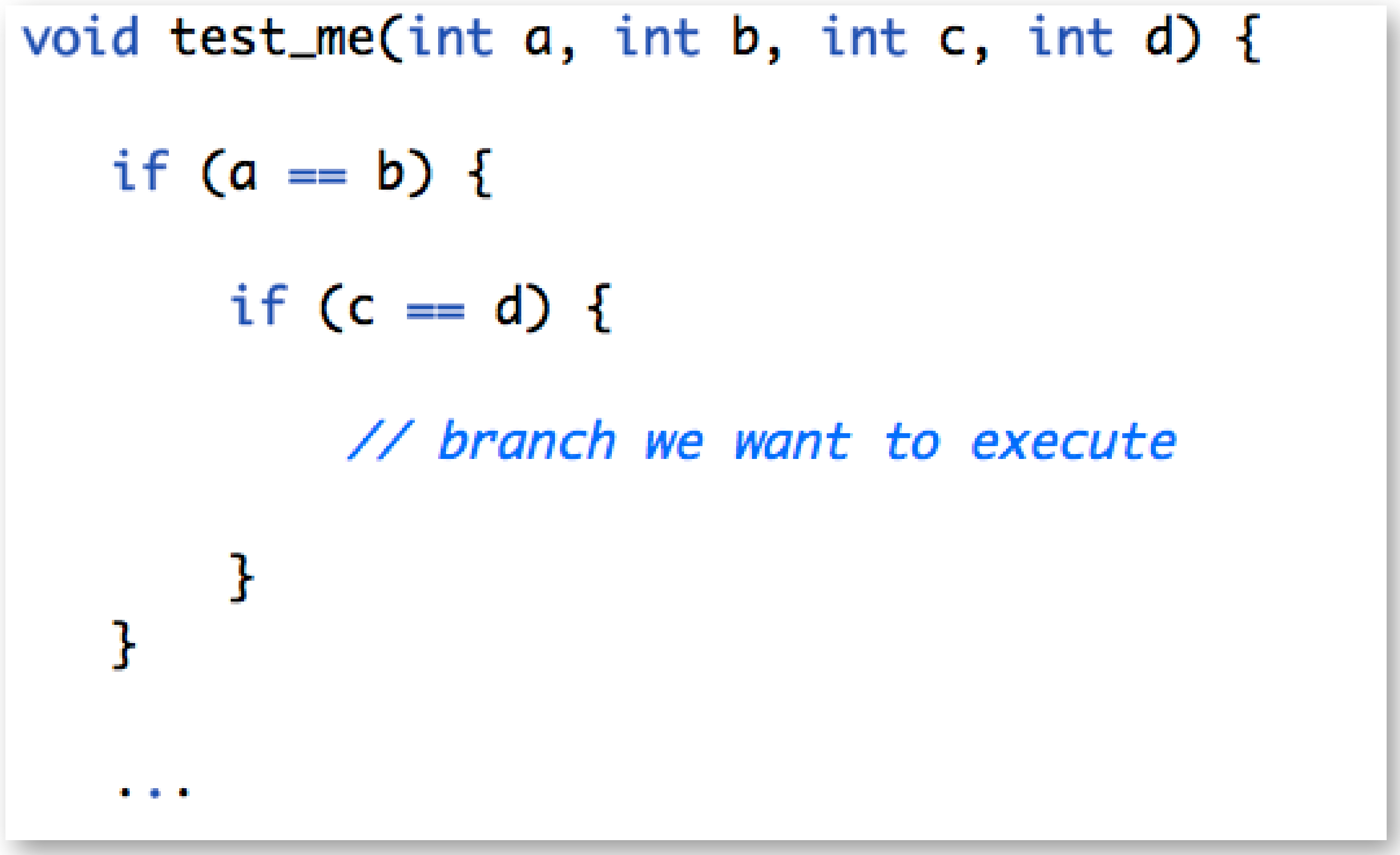
Crossover



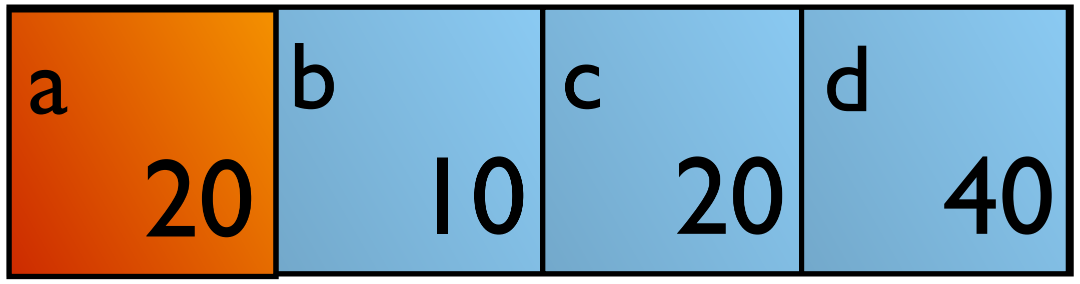
```
void test_me(int a, int b, int c, int d) {

   if (a == b) {

      if (c == d) {

         // branch we want to execute

      }
   }

   ...
```



Mutation



```
void test_me(int a, int b, int c, int d) {

  if (a == b) {

    if (c == d) {

      // branch we want to execute

    }
  }

  ...
```



Selection

- Selective pressure:
  The higher, the more likely the fittest are chosen
- Stagnation:
  Selective pressure too small
- Premature convergence:
  Selective pressure too high
- Standard algorithms:
  Rank selection, tournament selection, roulette wheel selection

EVOSUITE

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

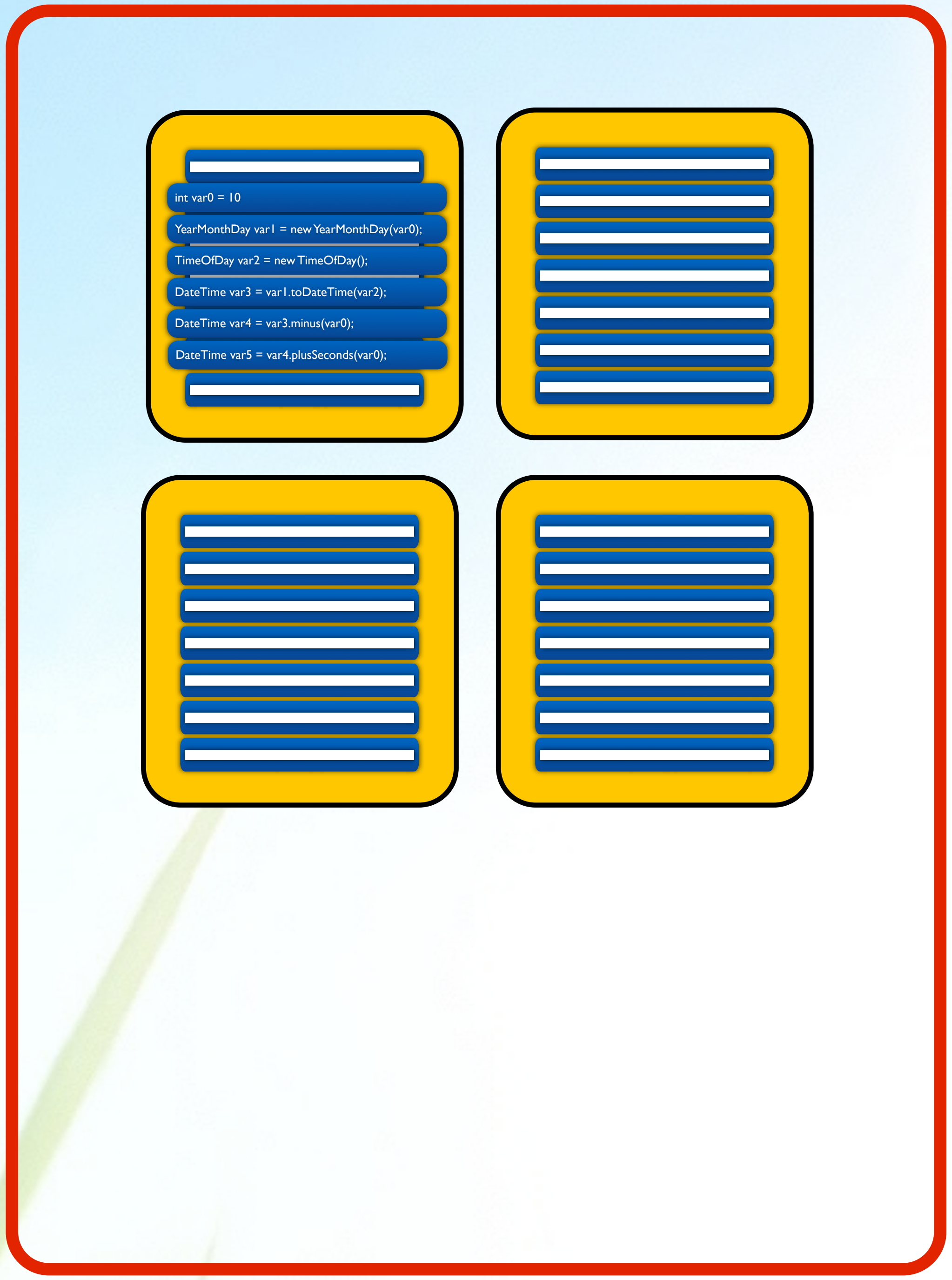
EVOSUITE

Outline

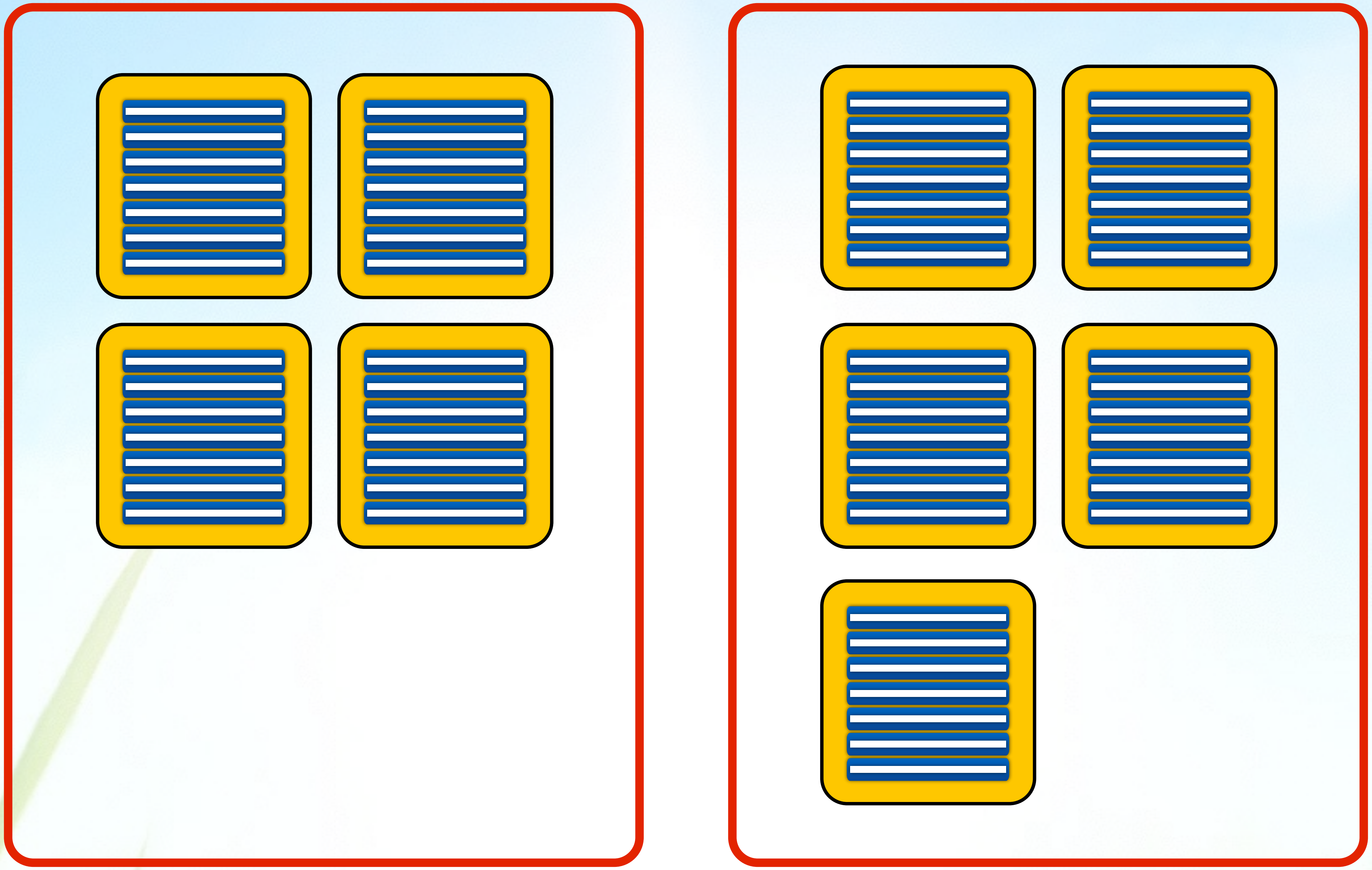
1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. **Generating Unit Tests with EvoSuite**
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

```
@Test

public void test()
{
    int x = 2;
    int y = 2;

    int result = x + y;

    assertEquals(4, result);

}
```

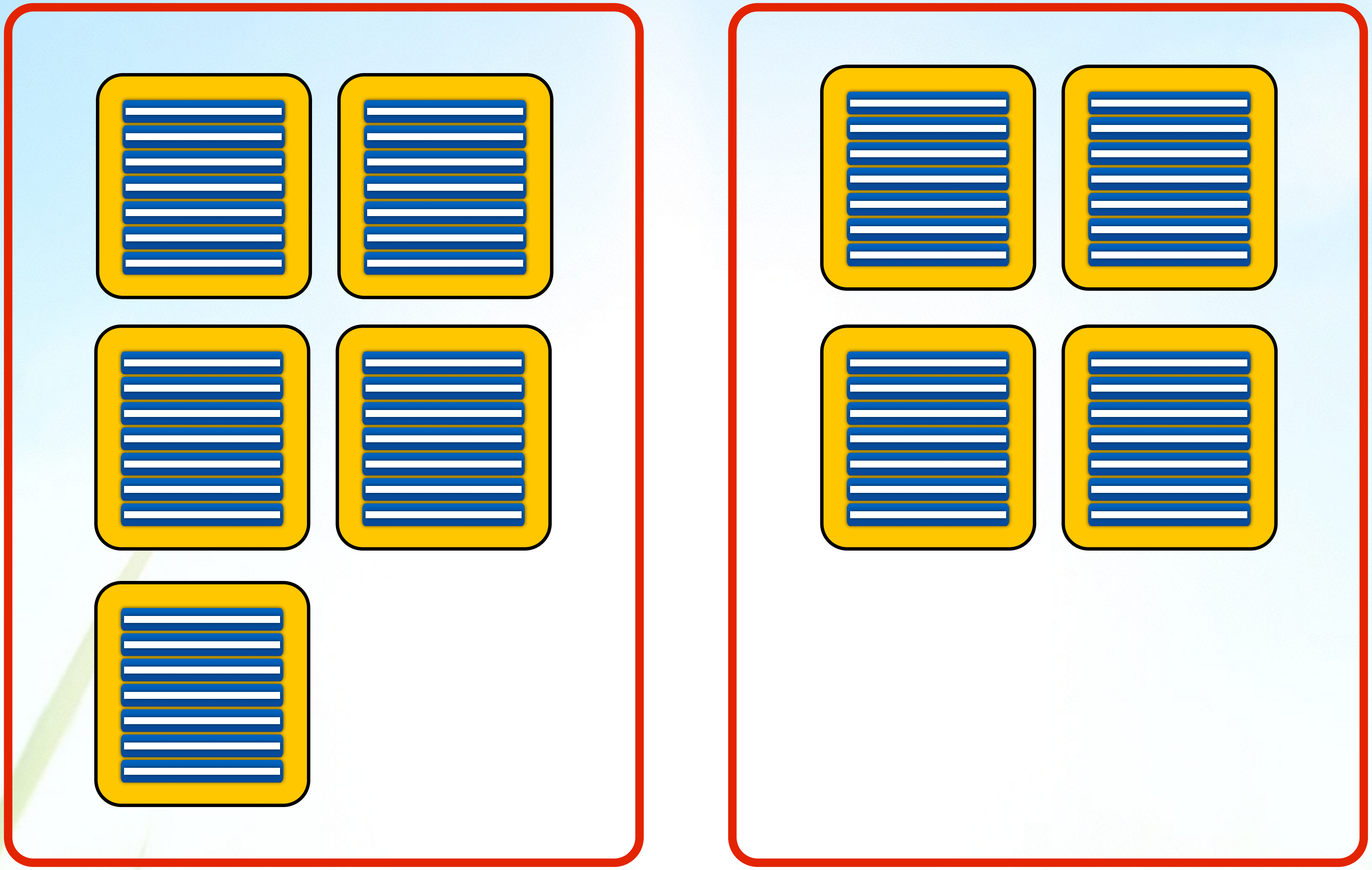
```
@Test
public void test()
{
    int var0 = 10
    YearMonthDay var1 = new YearMonthDay(var0);
    TimeOfDay var2 = new TimeOfDay();
    DateTime var3 = var1.toDateTime(var2);
    DateTime var4 = var3.minus(var0);
    DateTime var5 = var4.plusSeconds(var0);
}
```



```
int var0 = 10
YearMonthDay var1 = new YearMonthDay(var0);
TimeOfDay var2 = new TimeOfDay();
DateTime var3 = var1.toDateTime(var2);
DateTime var4 = var3.minus(var0);
DateTime var5 = var4.plusSeconds(var0);
```

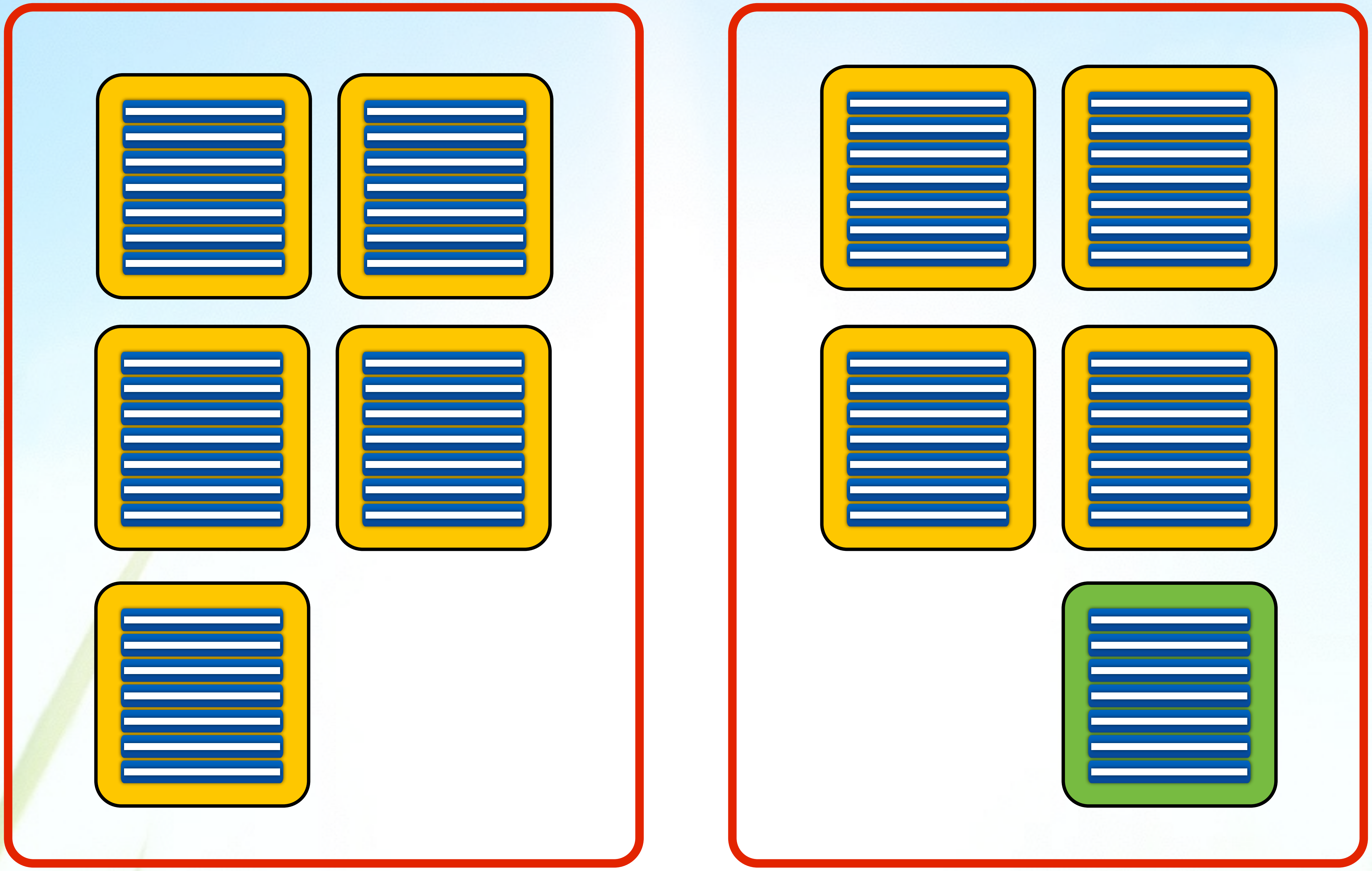


Crossover



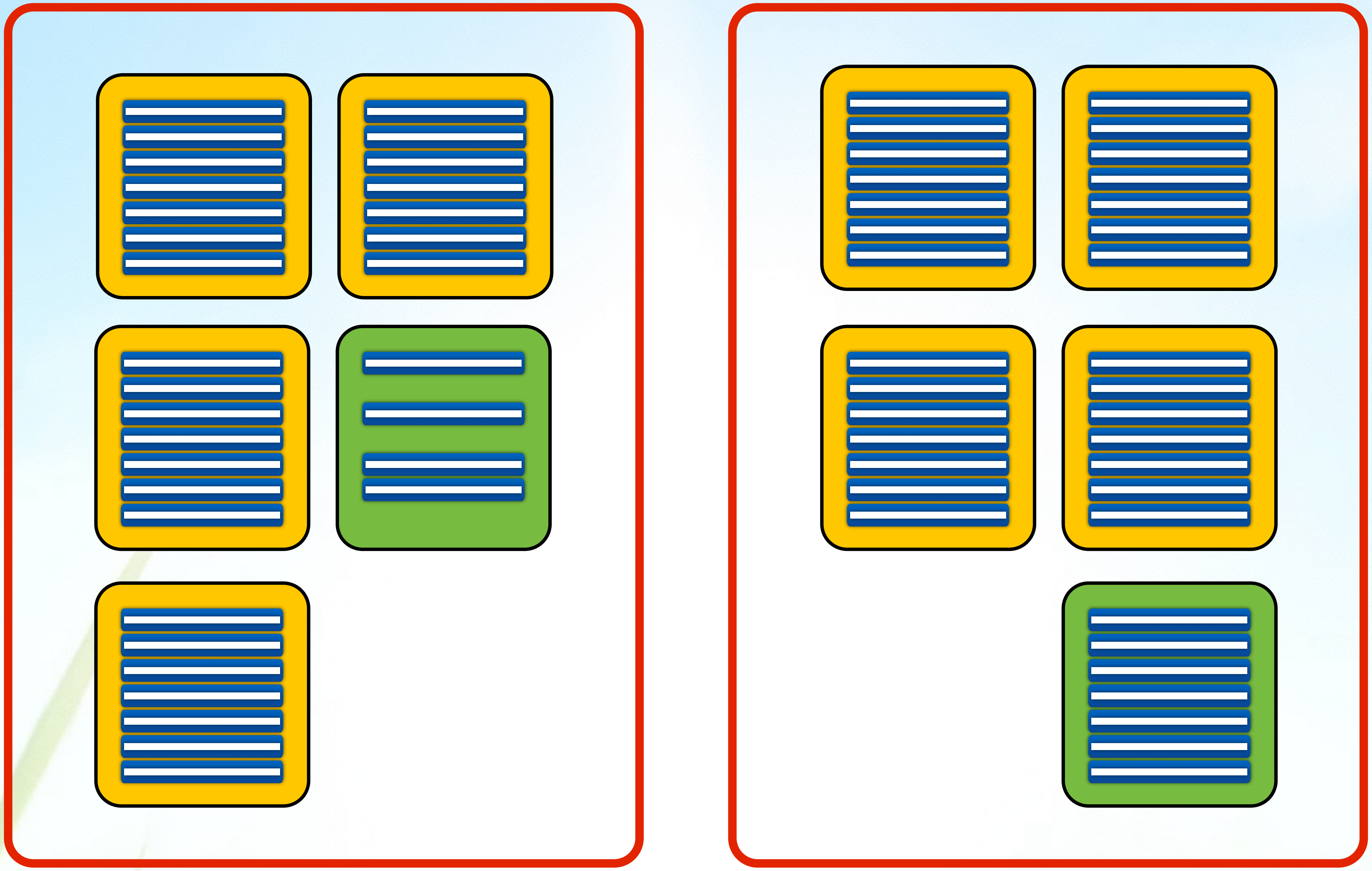
EVOSUITE

Mutation



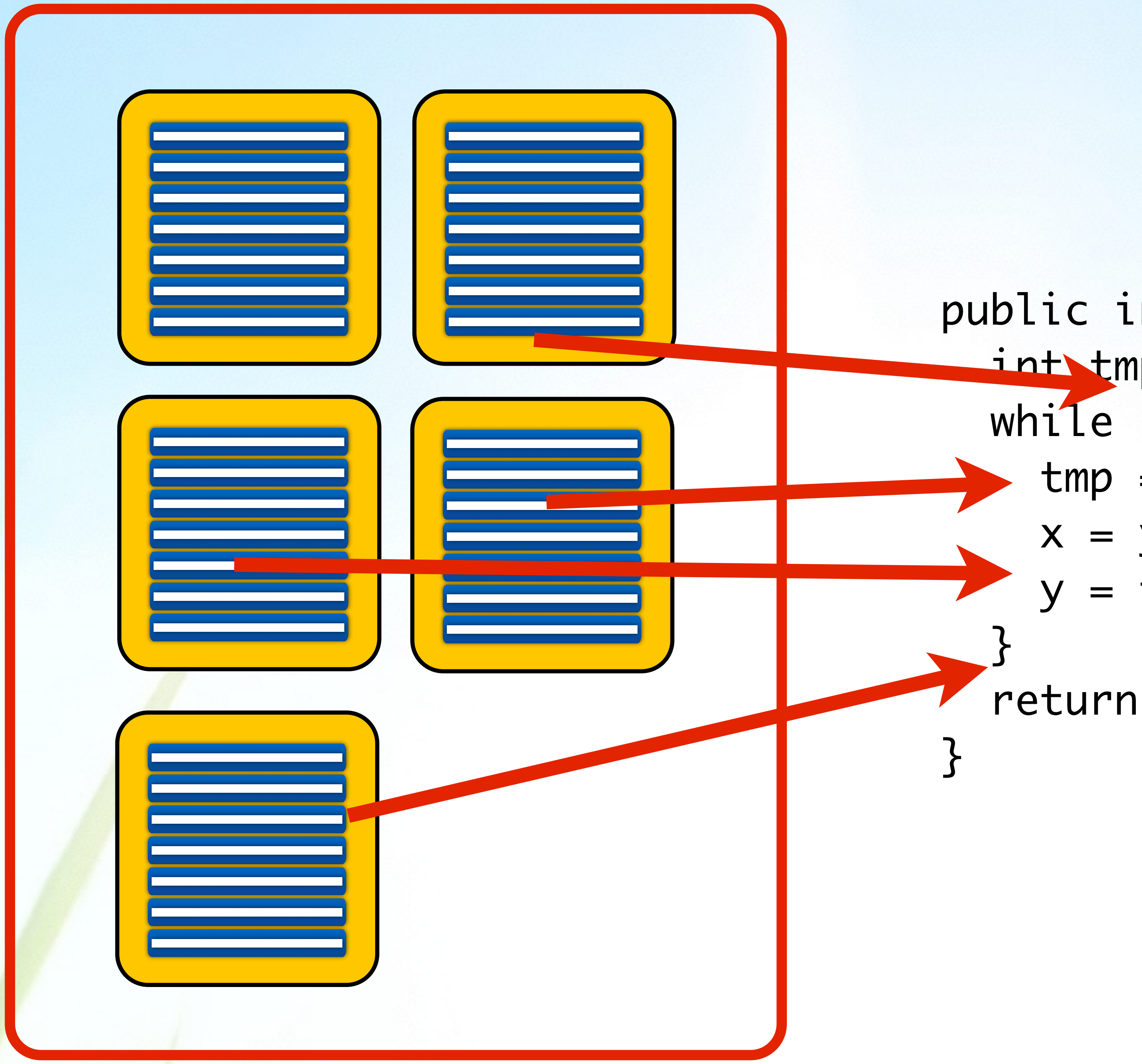
EVOSUITE

Mutation



EVOSUITE

Fitness

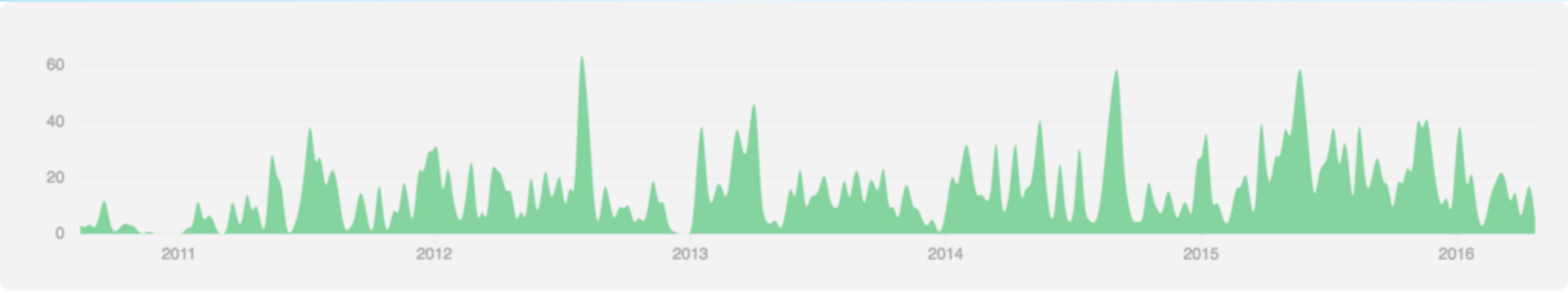


```
public int gcd(int x, int y) {
  int tmp;
  while (y != 0) {
    tmp = x % y;
    x = y;
    y = tmp;
  }
  return x;
}
```

Components of an SBST Tool

Search Algorithm	Genetic Algorithm (+Archive, Seeding, Local Search, DSE)
Representation	Sets of sequences of Java statements
Search Operators	Standard GA operators implemented for test suites
Fitness Function	Sum of branch distances (and others)
Test Execution	Java reflection
Instrumentation	Java bytecode instrumentation

Stats



- 6,865 commits
- 229,889 LOC
- 2,420 tests

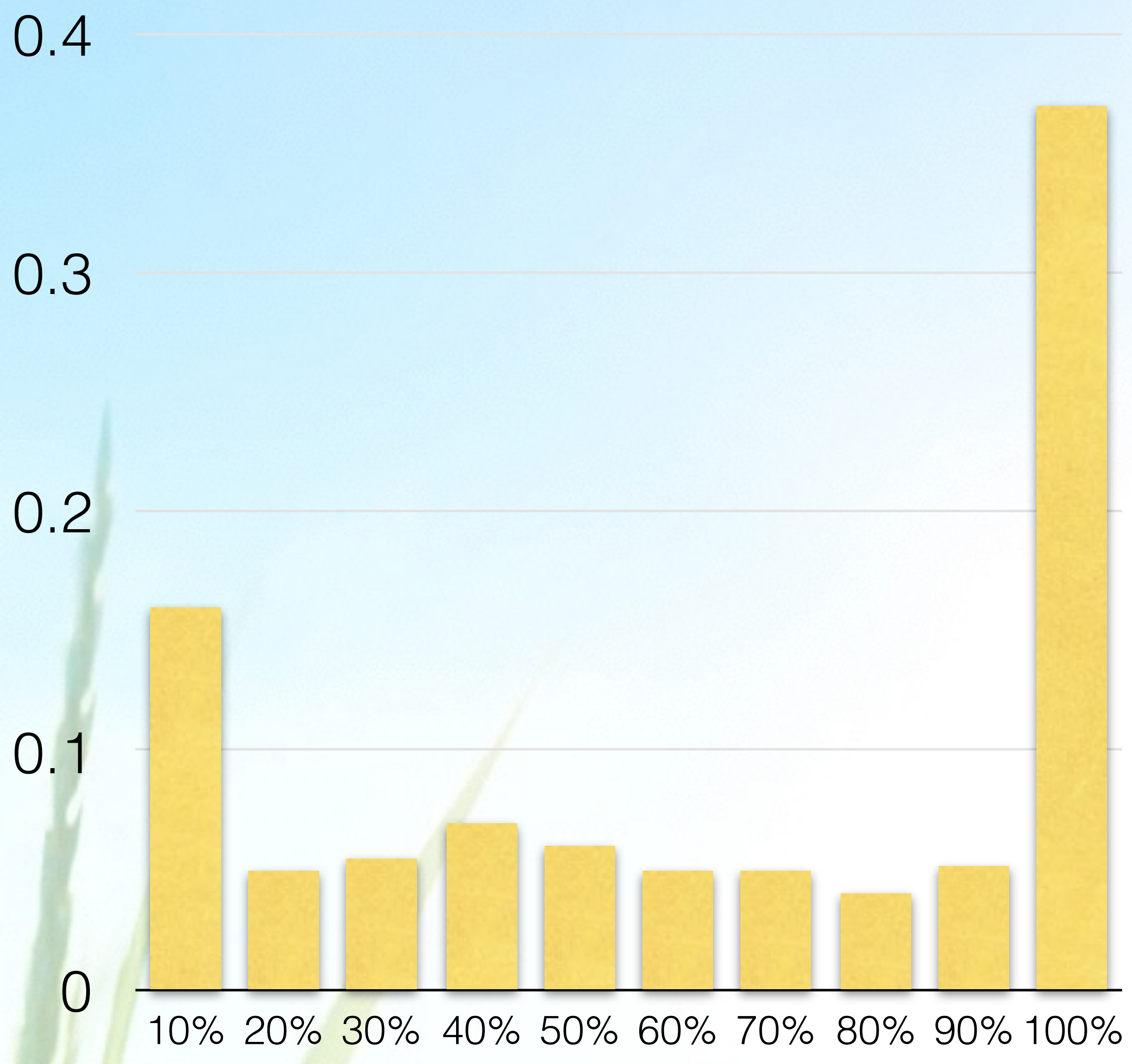
EVOSUITE

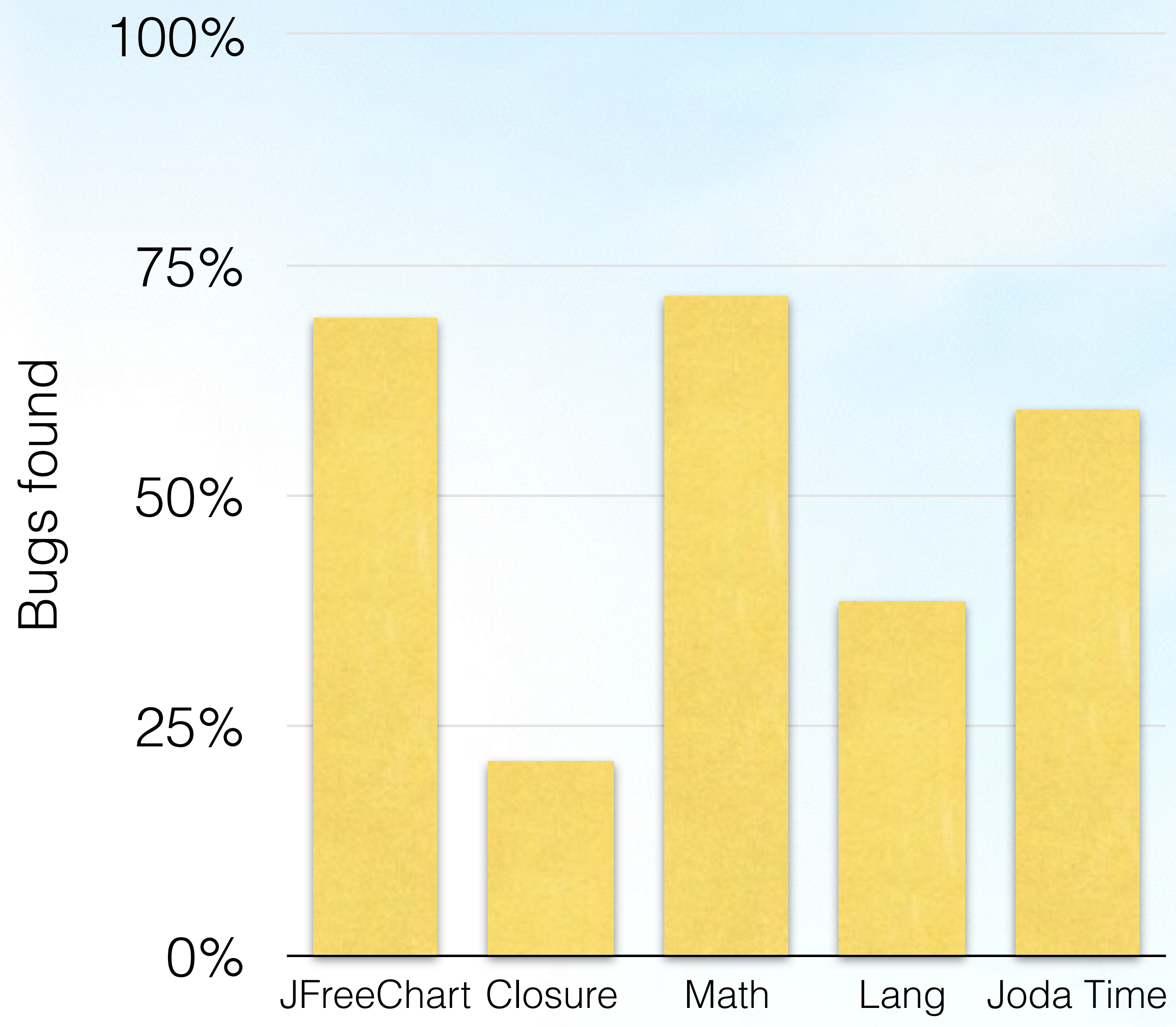
Acknowledgements

Andrea Arcuri
José Campos
Benjamin Friedrich
Florian Gross
Juan Pablo Galeotti
Alessandra Gorla
Mat Hall
Fitsum Meshesha Kifitew
Merlin Lang
Yanchuan Li
Eva May
Phil McMinn
Andre Mis
Daniel Muth
Annibale Panichella
David Paterson
Jeremias Roessler
Jose Miguel Rojas
Kaloyan Rusev
Sina Shamshiri
Sebastian Steenbuck
Andrey Tarasevich
Mattia Vivanti
Thomas White

EVOSUITE

Does it work?





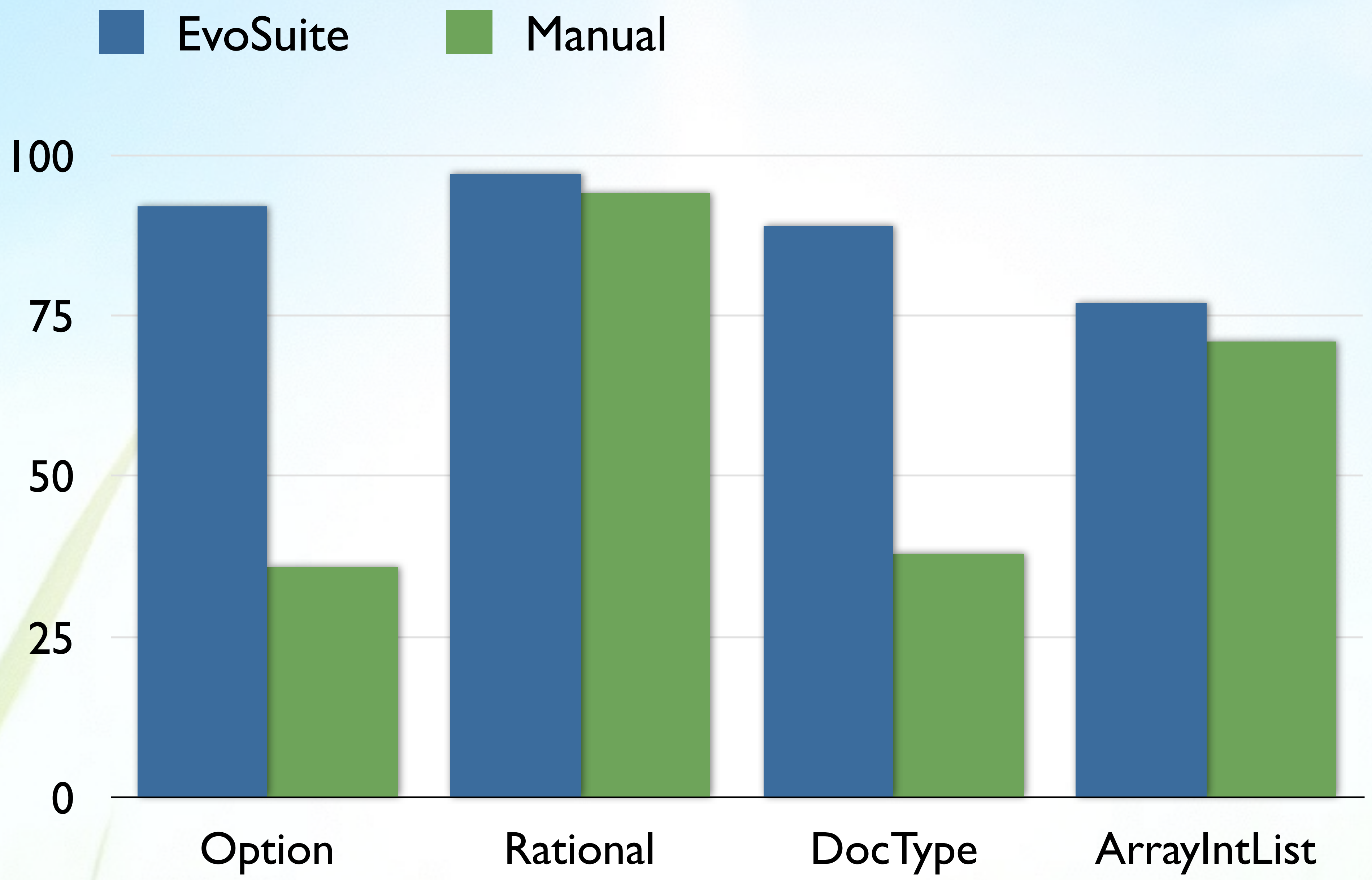
SF110: 23,886 Classes
6,628,619 LOC

Defects4J: 357 real bugs

G. Fraser, A. Arcuri. "A Large Scale Evaluation of Automated Unit Test Generation with EvoSuite" TOSEM 24(2), 2014.

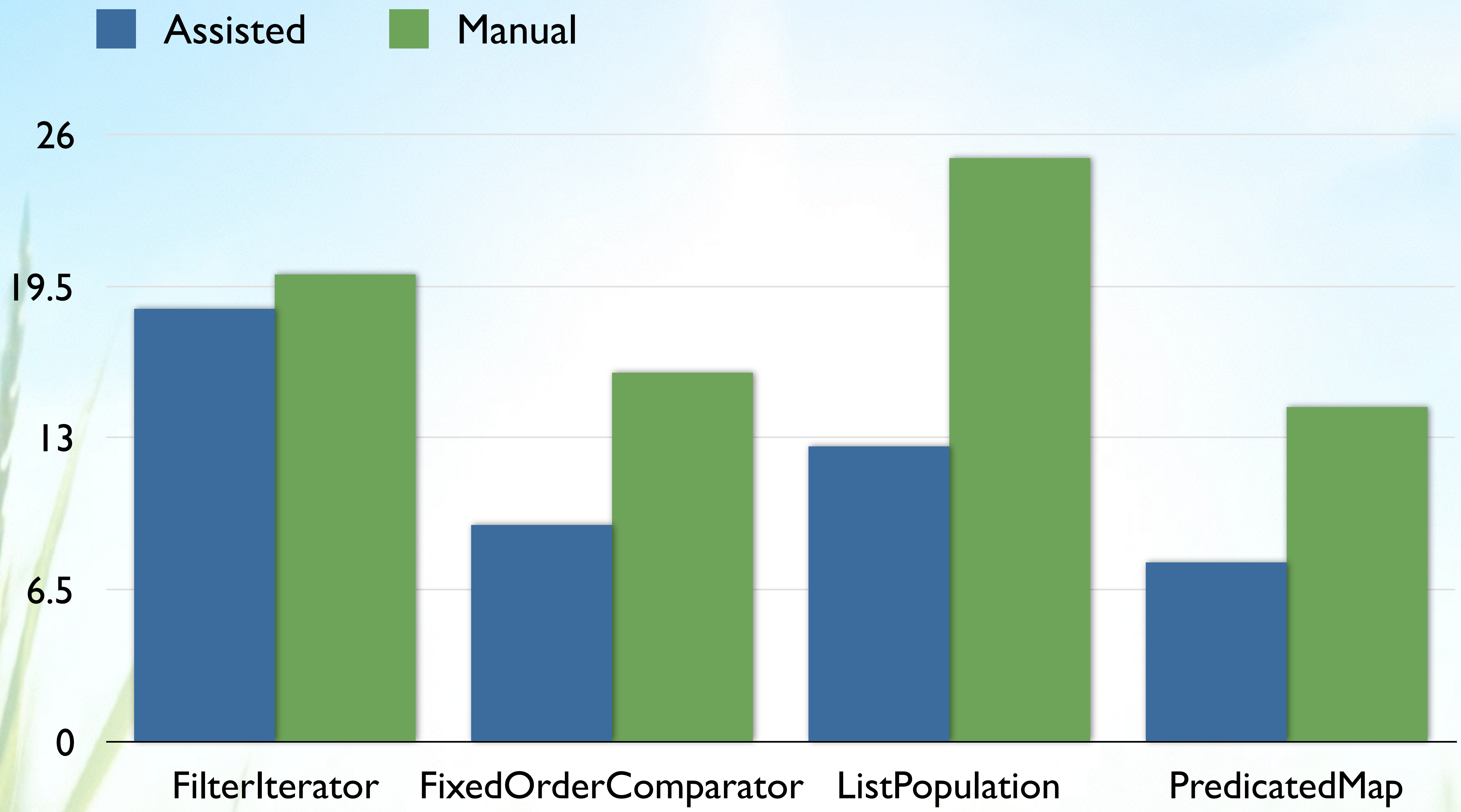
Shamshiri et al. "Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges" ASE, 2015

Coverage



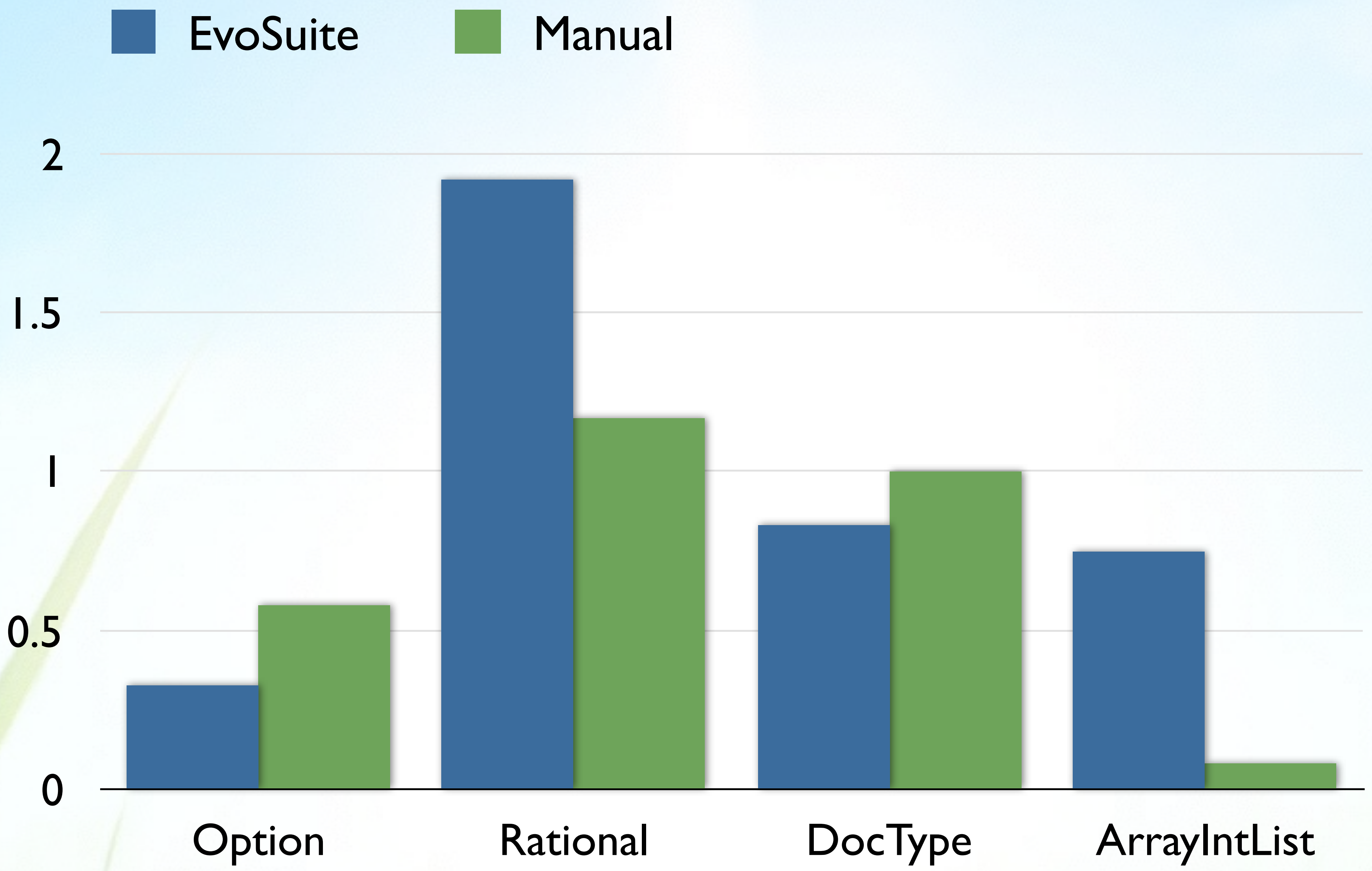
G. Fraser et al. "Does automated unit test generation really help software testers? A controlled empirical study." TOSEM, 2015

Time Spent on Testing



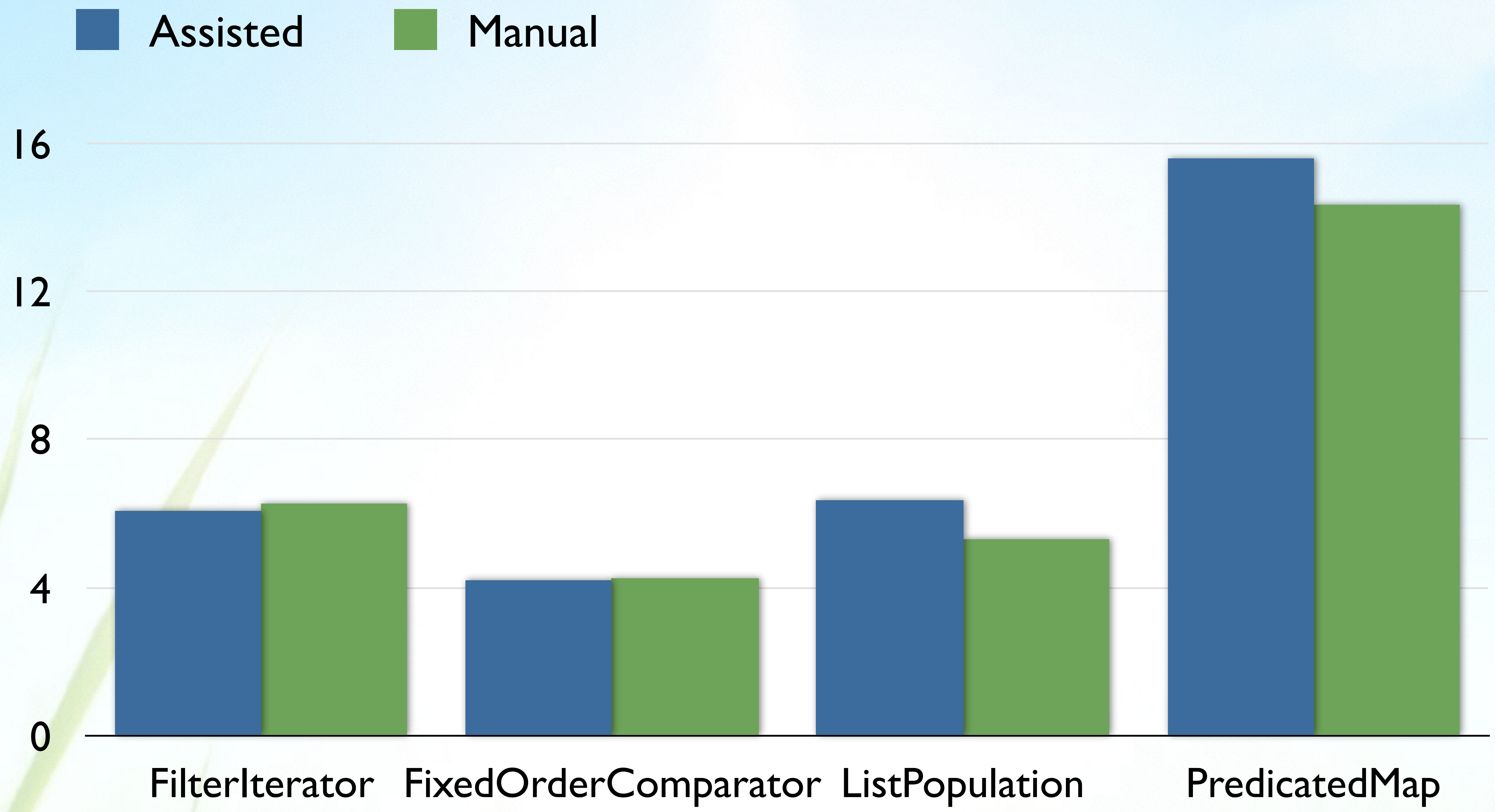
J. Rojas et al. "Automated unit test generation during software development: A controlled experiment and think-aloud observations." , ISSTA 2015

Fault Detection



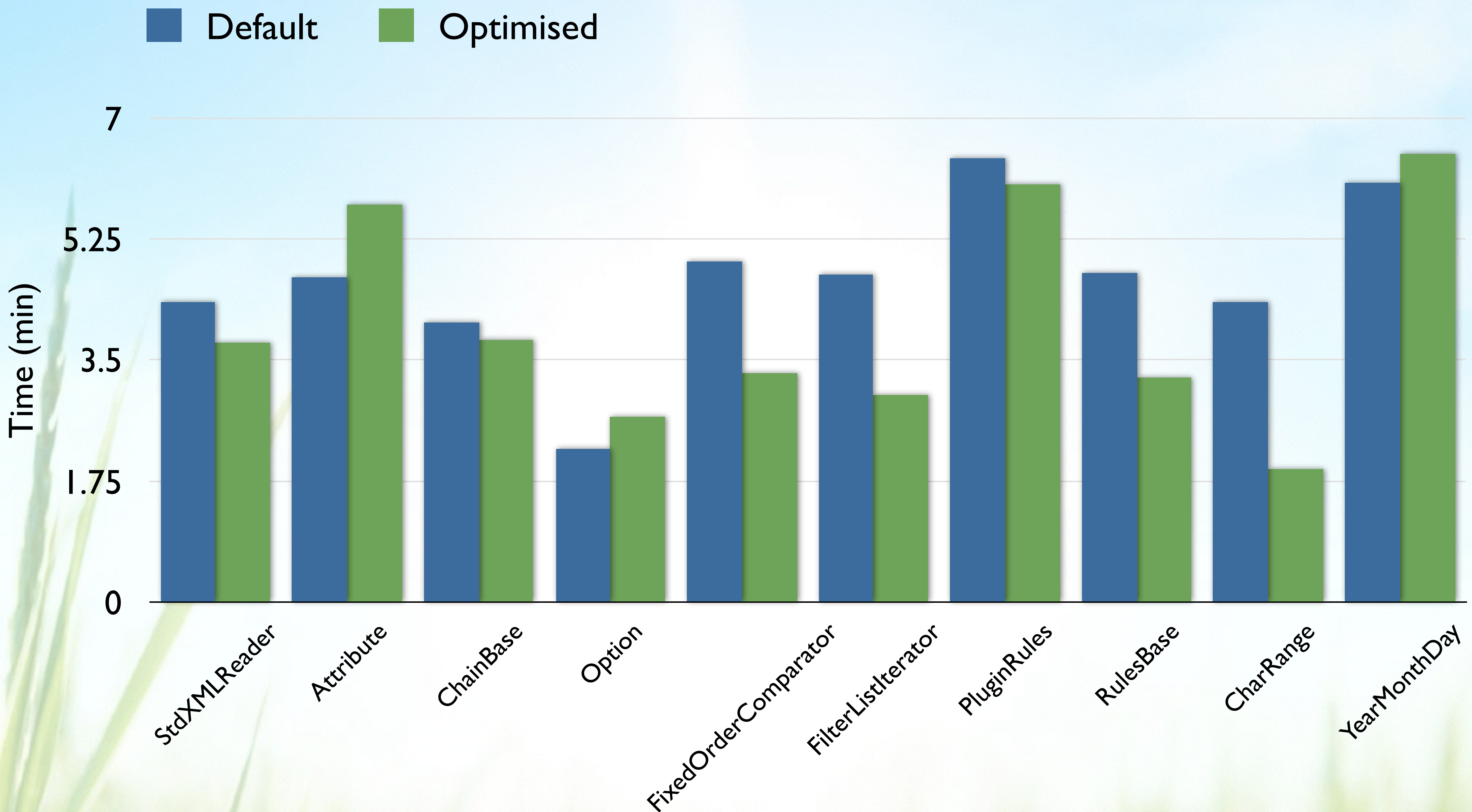
G. Fraser et al. "Does automated unit test generation really help software testers? A controlled empirical study." TOSEM, 2015

Faults Prevention



J. Rojas et al. "Automated unit test generation during software development: A controlled experiment and think-aloud observations." , ISSTA 2015

Time Spent Understanding



Method Names

```
@Test(timeout = 4000)
public void test3() throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```

```
@Test(timeout = 4000)
public void testFooReturningFalse() throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```

EVOSUITE

Variable Names

```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```

```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample invokesFoo = new StringExample();
    boolean resultFromFoo = invokesFoo.foo("");
    assertFalse(resultFromFoo);
}
```

Variable Names

```
public class Foo {
  public void foo() {
    StringExample sx = new StringExample();
    boolean bar = sx.foo("");
  }
}
```

```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample sx = new StringExample();
    boolean bar = sx.foo("");
    assertFalse(bar);
}
```

EVOSUITE

Getting EvoSuite

`http://www.evosuite.org/downloads`

- Jar release - for command line usage
- Maven plugin
- IntelliJ plugin
- Eclipse plugin
- Jenkins plugin

Testing a Class

- Demo - command line
- Main options:
  ```
  -projectCP
  -class
  -criterion
  ```

Properties

- `-Dproperty=value`
- Search budget (s)
  `-Dsearch_budget=60`
- Assertion generation
  `-Dassertions=false`
  `-Dassertion_strategy=all`
- Minimisation (length and values)
  `-Dminimize=false`
- Inlining
  `-Dinline=false`

EVOSUITE

EvoSuite Sandbox

- Demo - Nondeterministic class
- Runtime library to execute tests

Testing multiple classes

Demo:

- Target / prefix
- Continuous
- Maven
- Jenkins
- IntelliJ

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. **Generating Unit Tests with EvoSuite**
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

Outline

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to test my own Java code?
- Yes, of course

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to implement my ideas on unit test generation?
- Yes, of course

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to study developer behaviour?
- Yes, of course

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to generate unit tests for my experiment on X?
- Yes, of course

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to build a unit test generator for a different programming language?
- EvoSuite is 90% JVM handling code
- Would need to reimplement representation, search operators, fitness functions, test execution, …

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to create an Android testing tool?
- Android uses Java / Dalvik bytecode
- Can also compile to Java bytecode
- How to handle Android dependencies?

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to create a GUI testing tool?
- If you want to test Java/Swing applications…
- But a GA may not be the right choice

EVOSUITE

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to create a web app testing tool?
- If it's based on JEE, unit testing already works (JEE support is not complete yet)
- System testing…see GUI testing

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to implement a non-test generation SBSE tool?
- GA implementation is quite test specific
- Using for other purposes would need refactoring
  But then, is it better than using existing generic GA libraries?
- If the tool uses Java, why not?

When to use and not to use EvoSuite

- Should I use EvoSuite…
- …to implement a tool that requires tests?
- E.g., specification mining, fault localisation, program repair, GI, …
- Sure, integrating EvoSuite should be easy

Outline

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work in unit test generation

Building EvoSuite

- Git repository:

```
git clone https://github.com/EvoSuite/evosuite.git
```

- Maven

```
mvn package
(mvn -DskipTests package)
```

- Where is EvoSuite now?

```
master/target/evosuite-master-1.0.4-SNAPSHOT.jar
```

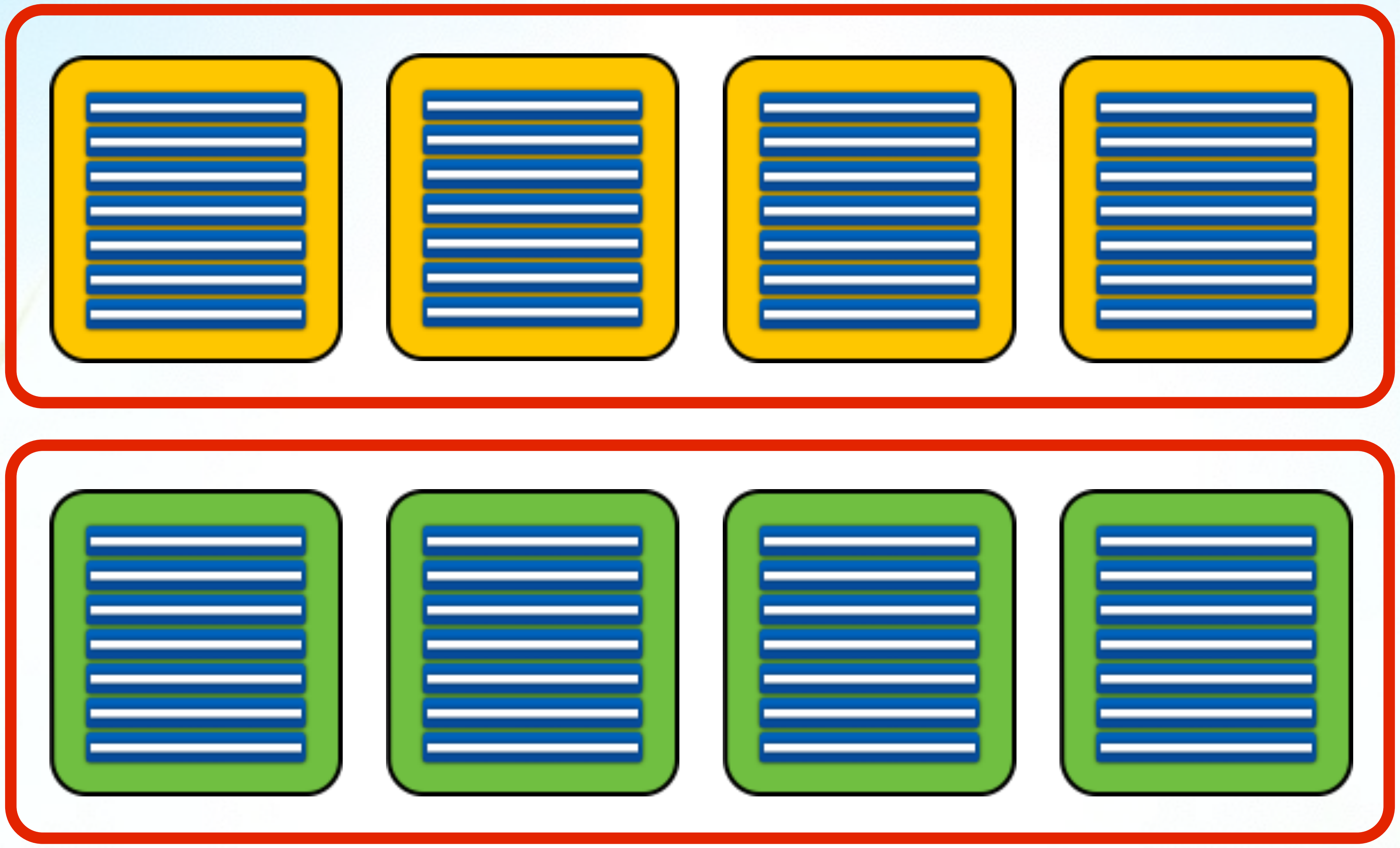
- Why is the jar file so huge?

Module Structure

- master
- client
- runtime
- standalone-runtime
- plugins
- generated
- shaded

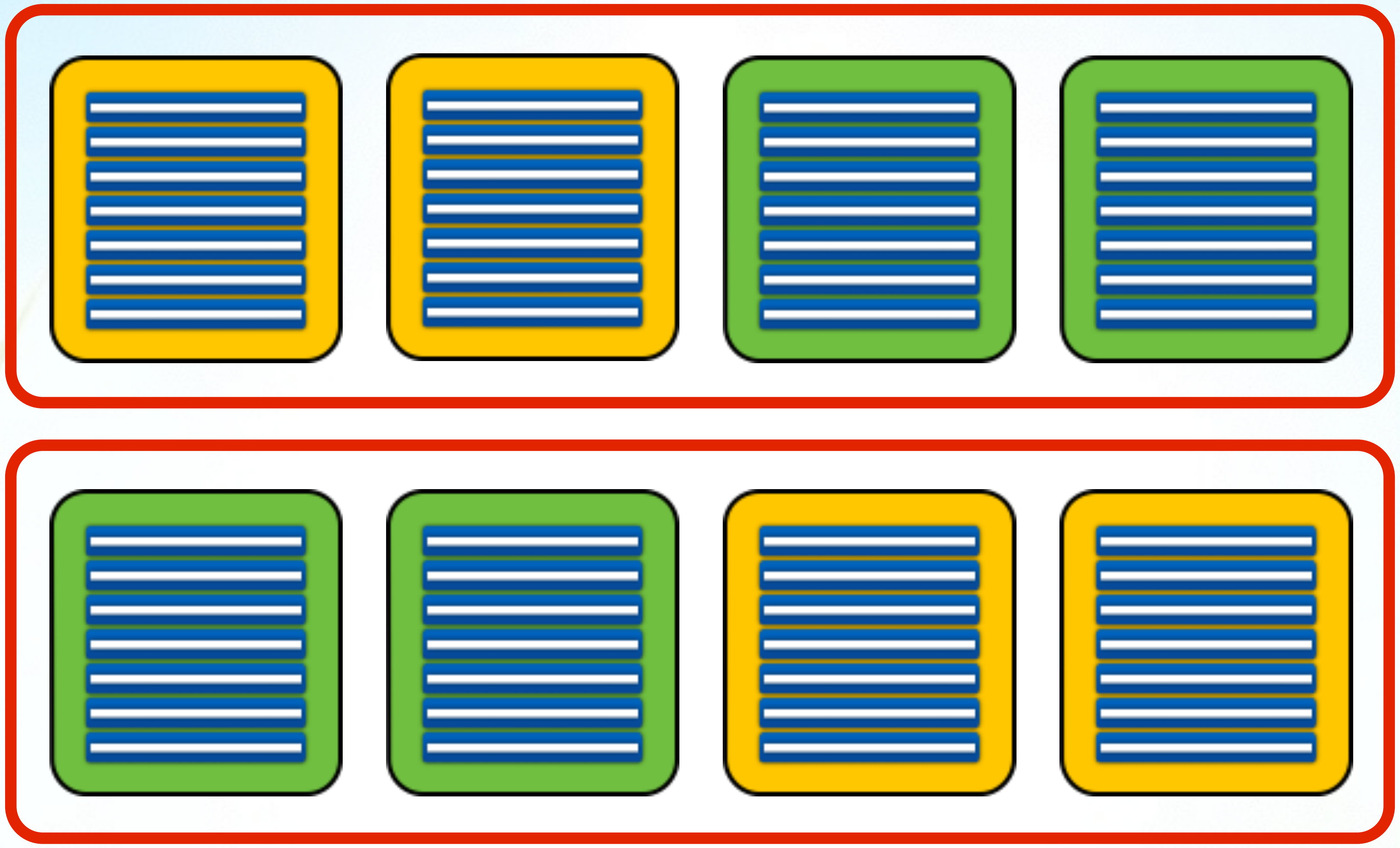
Extending EvoSuite

- (Artificial) Example: Middle point crossover



Extending EvoSuite

- (Artificial) Example: Middle point crossover



EVOSUITE

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. **Extending EvoSuite**
6. Ideas for future work

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work

1. SBST is Slow

- Fitness evaluation means executing tests
- Executing tests is slow
- How to reduce the number of fitness evaluations?
- How to improve search operators?
- Can we use ML to predict test execution results?

EVOSUITE

2. OO Guidance

- Object oriented code has a terrible search landscape
- Complex dependency objects are a problem
- Include dependency objects in fitness functions?
- Better testability transformations?
- Better fitness functions?

3. New Features

- Integration testing
- Concurrent code
- GUI handling code
- Database dependent code
- Prioritising tests

4. SBST Usability

- Assertion/contract testing code?
- Coverage isn't a great objective
- Usability as optimisation goal
- Study developers using SBST tools

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work

Outline

1. What is Search Based Software Testing?
2. Building an SBST Tool is Easy!
3. Generating Unit Tests with EvoSuite
4. When to use and not to use EvoSuite
5. Extending EvoSuite
6. Ideas for future work

Online Tutorials

- Using EvoSuite on the command line:
  `http://www.evosuite.org/documentation/tutorial-part-1/`
- Using EvoSuite with Maven:
  `http://www.evosuite.org/documentation/tutorial-part-2/`
- Running experiments with EvoSuite:
  `http://www.evosuite.org/documentation/tutorial-part-3/`
- Extending EvoSuite:
  `http://www.evosuite.org/documentation/tutorial-part-4/`

2. Corner Cases

- Constant Seeding: +5%
- Virtual FS: +1.4%
- Mocking +4.7%
- JEE support: +3%
- DSE: +1.2%

3. Developers

```
public class Example {

    private Example() {}

    // …
}
```

4. Testing

EvoSuite uses one central random number generator

Any change will affect something at a completely different part of the program

Change seeds frequently during testing to find flaky tests