



EV SUITE TUTORIAL

Gordon Fraser, University of Sheffield

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

Why should you care?

- **Are you writing Java code?**
Learn how to use EvoSuite to help you testing
- **Are you doing research on testing?**
Learn how to use EvoSuite for experiments
Learn how to extend EvoSuite
- **Are you doing research on SBST?**
Hear about our experiences in developing an SBST tool

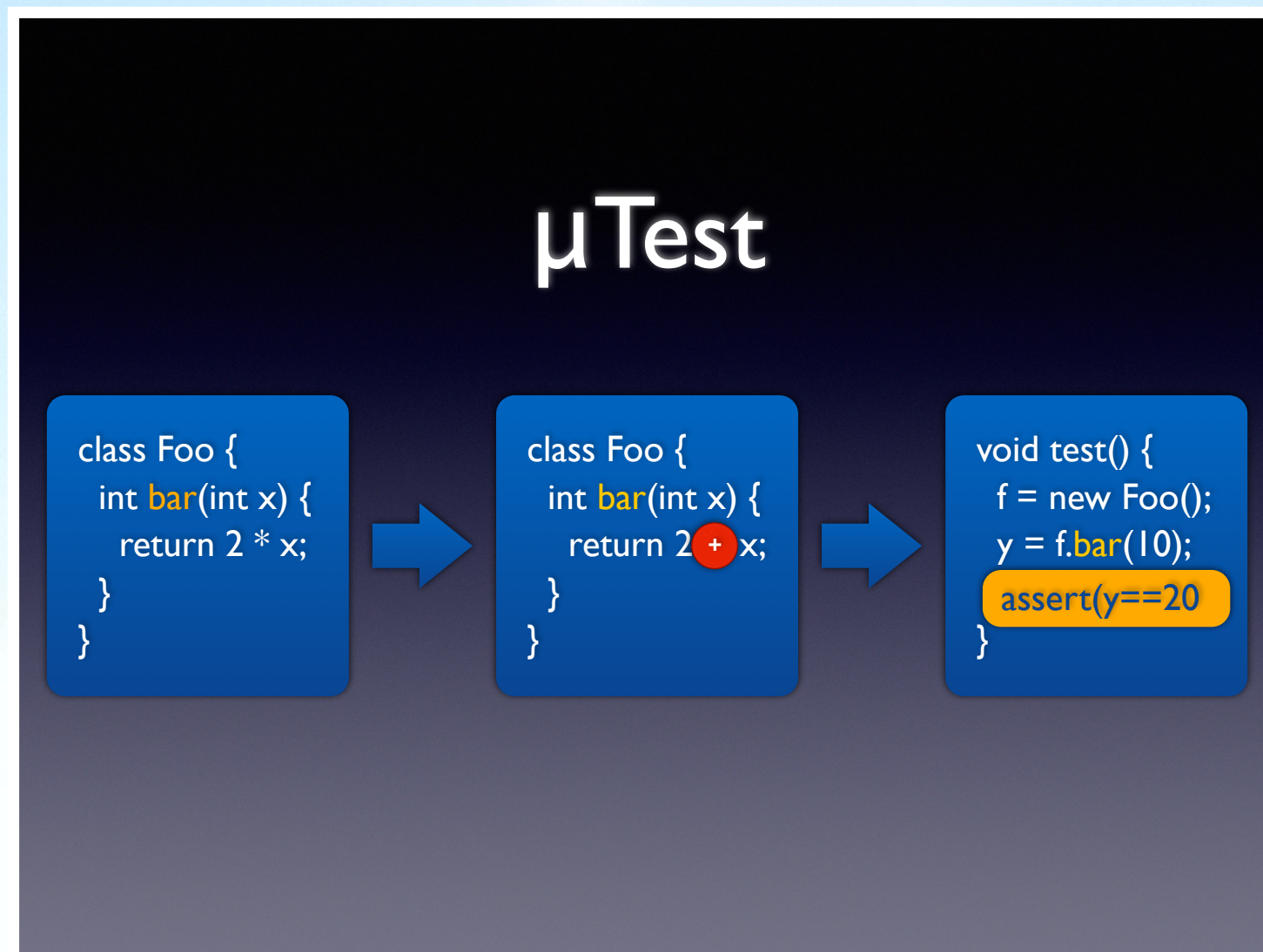
Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

History



Mutation-based Generation of Tests and Oracles, ISSTA 2010


```
LocalDate date = new LocalDate(2010, 7, 15);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2010);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getYear(), 2010);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
```

```
LocalDate date = new LocalDate(2010, 7, 15);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2010);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getYear(), 2010);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
```



```
assertEquals(date.getDayOfMonth(), ...);  
date.plusYears(1);  
assertEquals(date.getYear(), 2011);  
assertEquals(date.size(), 3);  
assertEquals(date.getValue(YEAR), 2011);  
assertEquals(date.getValue(MONTH_OF_YEAR), 7);  
assertEquals(date.getValue(DAY_OF_MONTH), 15);  
assertEquals(date.getLocalMillis(), ...);  
assertEquals(date, date);  
assertEquals(date.compareTo(date), 0);  
assertEquals(date.getYearOfEra(), ...);  
assertEquals(date.getYearOfCentury(), ...);  
assertEquals(date.getWeekyear(), ...);  
assertEquals(date.getMonthOfYear(), 7);  
assertEquals(date.getWeekOfWeekyear(), ...);  
assertEquals(date.getDayOfWeek(), ...);  
assertEquals(date.getDayOfMonth(), ...);
```



```
assertEquals(date.getDayOfMonth(), ...);  
date.plusYears(1);  
assertEquals(date.getYear(), 2011);  
assertEquals(date.size(), 3);  
assertEquals(date.getValue(YEAR), 2011);  
assertEquals(date.getValue(MONTH_OF_YEAR), 7);  
assertEquals(date.getValue(DAY_OF_MONTH), 15);  
assertEquals(date.getLocalMillis(), ...);  
assertEquals(date, date);  
assertEquals(date.compareTo(date), 0);  
assertEquals(date.getYearOfEra(), ...);  
assertEquals(date.getYearOfCentury(), ...);  
assertEquals(date.getWeekyear(), ...);  
assertEquals(date.getMonthOfYear(), 7);  
assertEquals(date.getWeekOfWeekyear(), ...);  
assertEquals(date.getDayOfWeek(), ...);  
assertEquals(date.getDayOfMonth(), ...);
```



```
LocalDate date = new LocalDate(2010, 7, 15);  
date.plusYears(1);  
assertEquals(date.getYear(), 2011);  
assertEquals(date.getValue(YEAR), 2011);
```



```
LocalDate date = new LocalDate(2010, 7, 15);  
date.plusYears(1);  
assertEquals(date.getYear(), 2011);
```


History



April 9, 2010

“Evolutionary Generation of Whole Test Suites,”
11th International Conference on Software Quality (QSIC 2011)

Stats



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


```
@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);

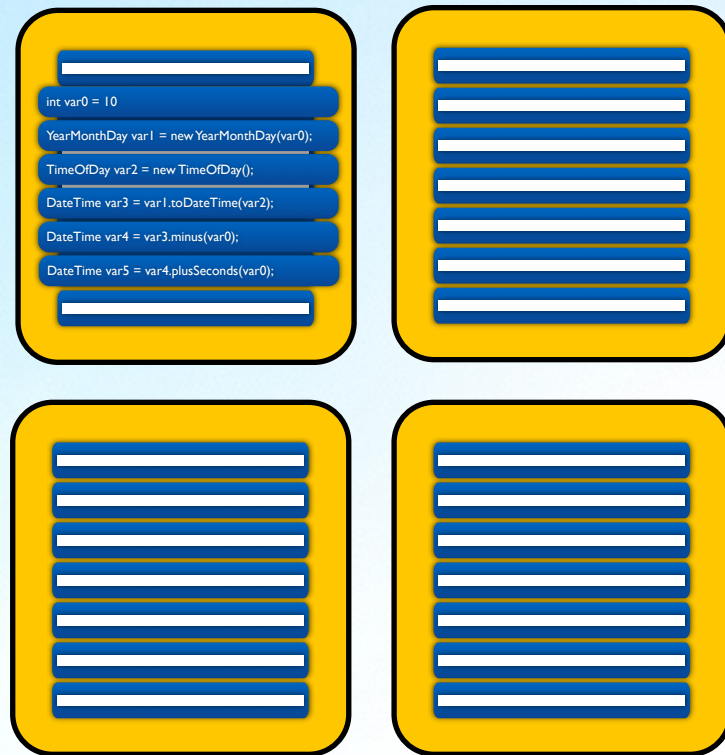
TimeOfDay var2 = new TimeOfDay();

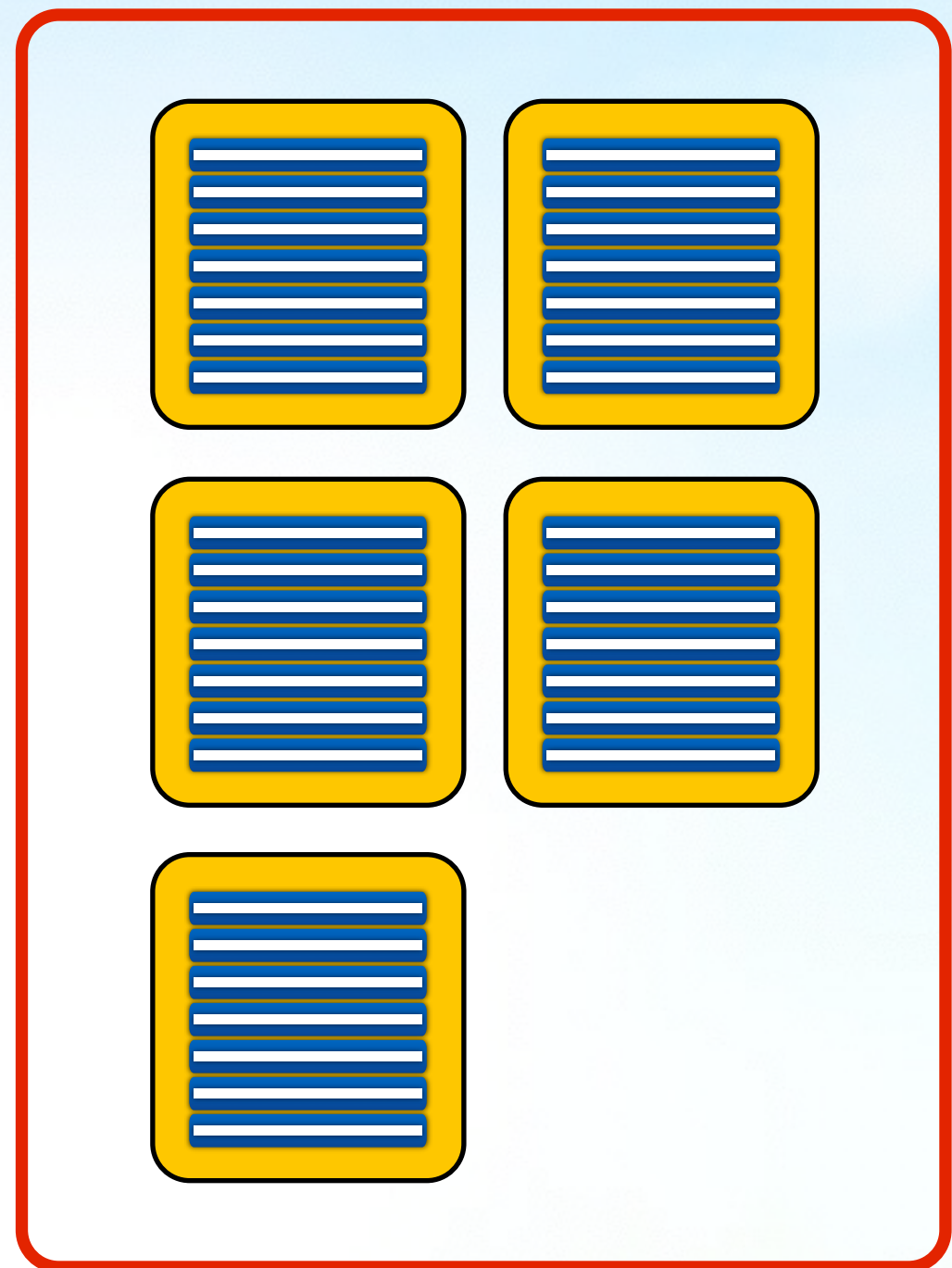
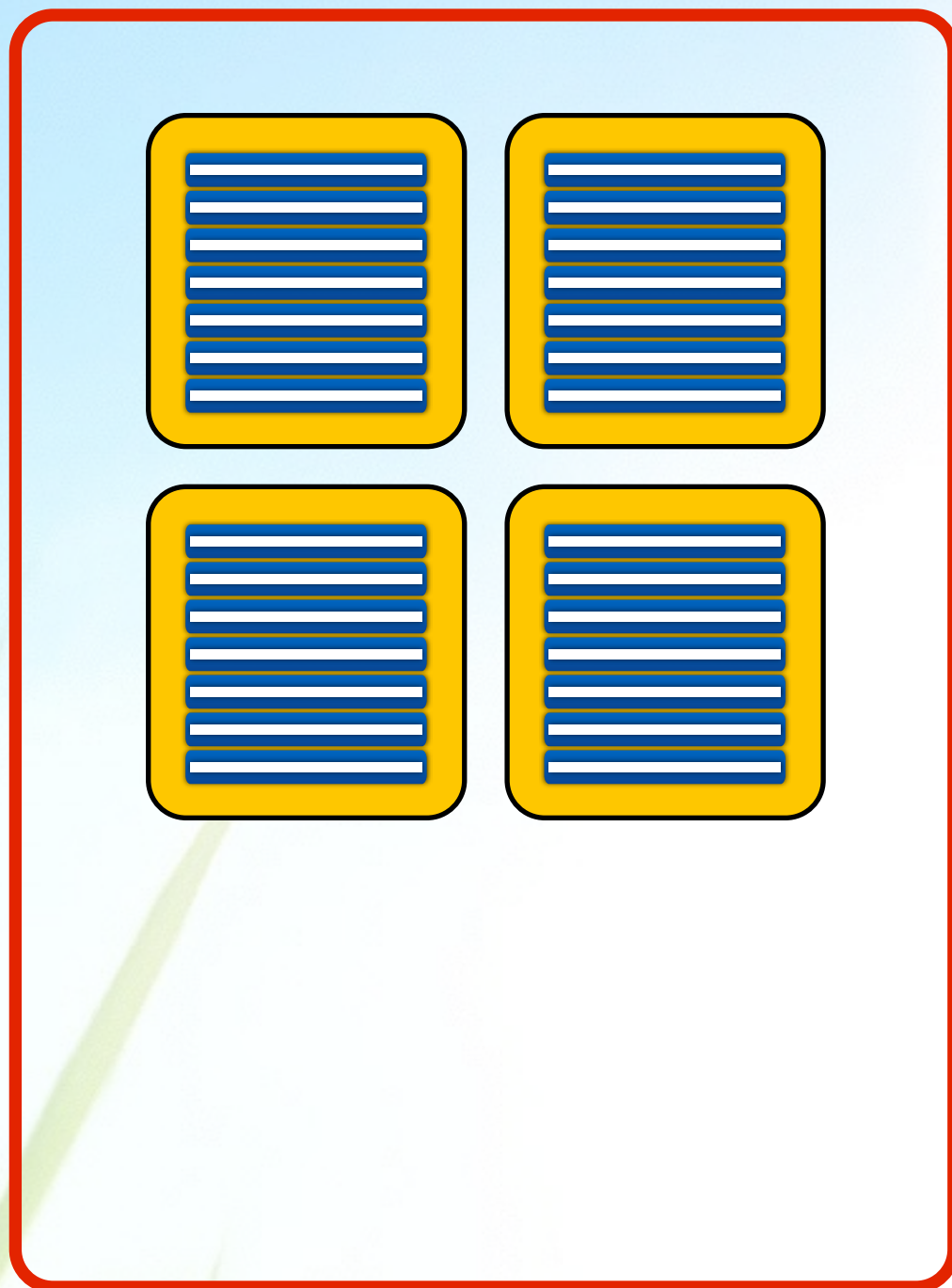
DateTime var3 = var1.toDateTime(var2);

DateTime var4 = var3.minus(var0);

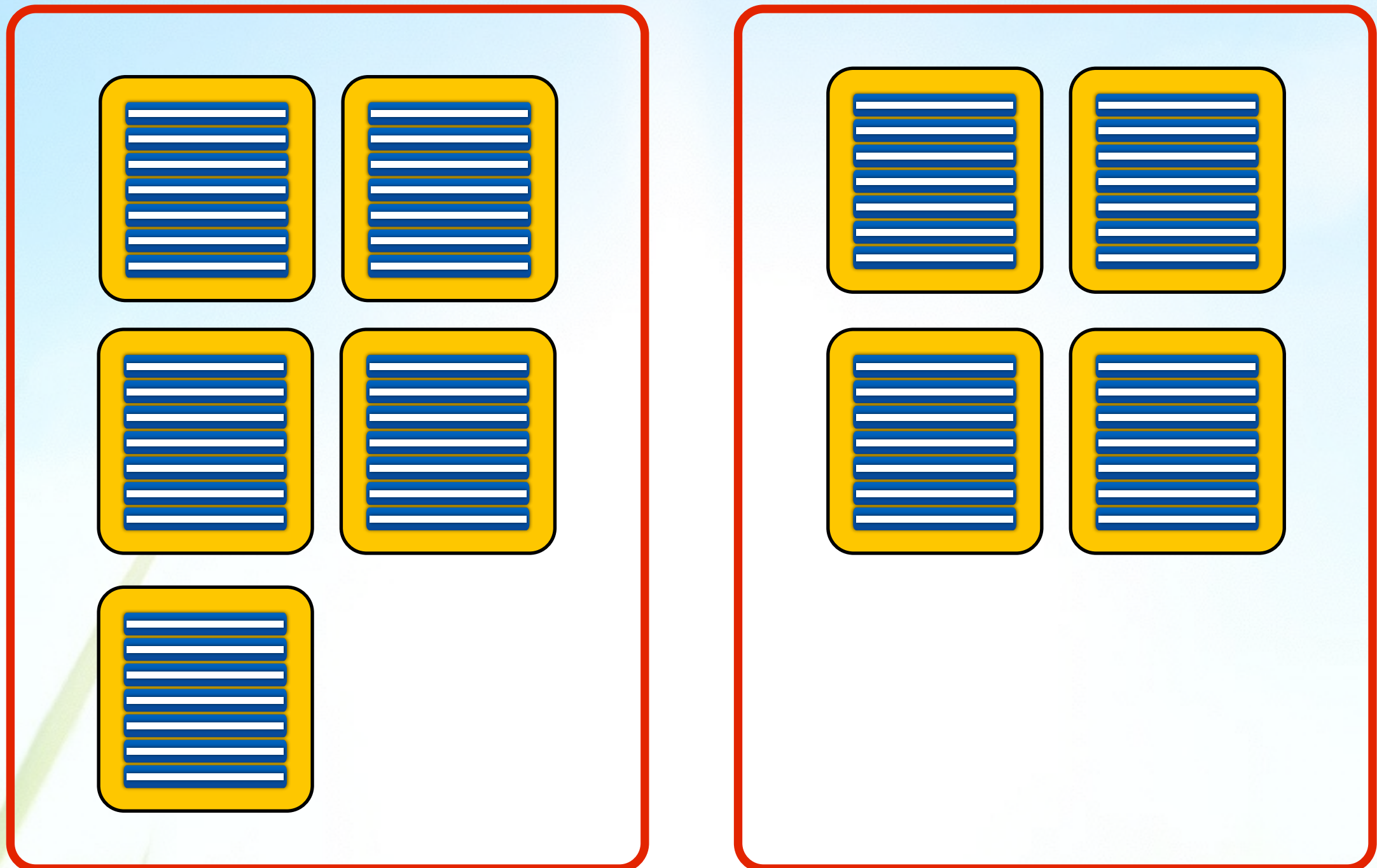
DateTime var5 = var4.plusSeconds(var0);

}

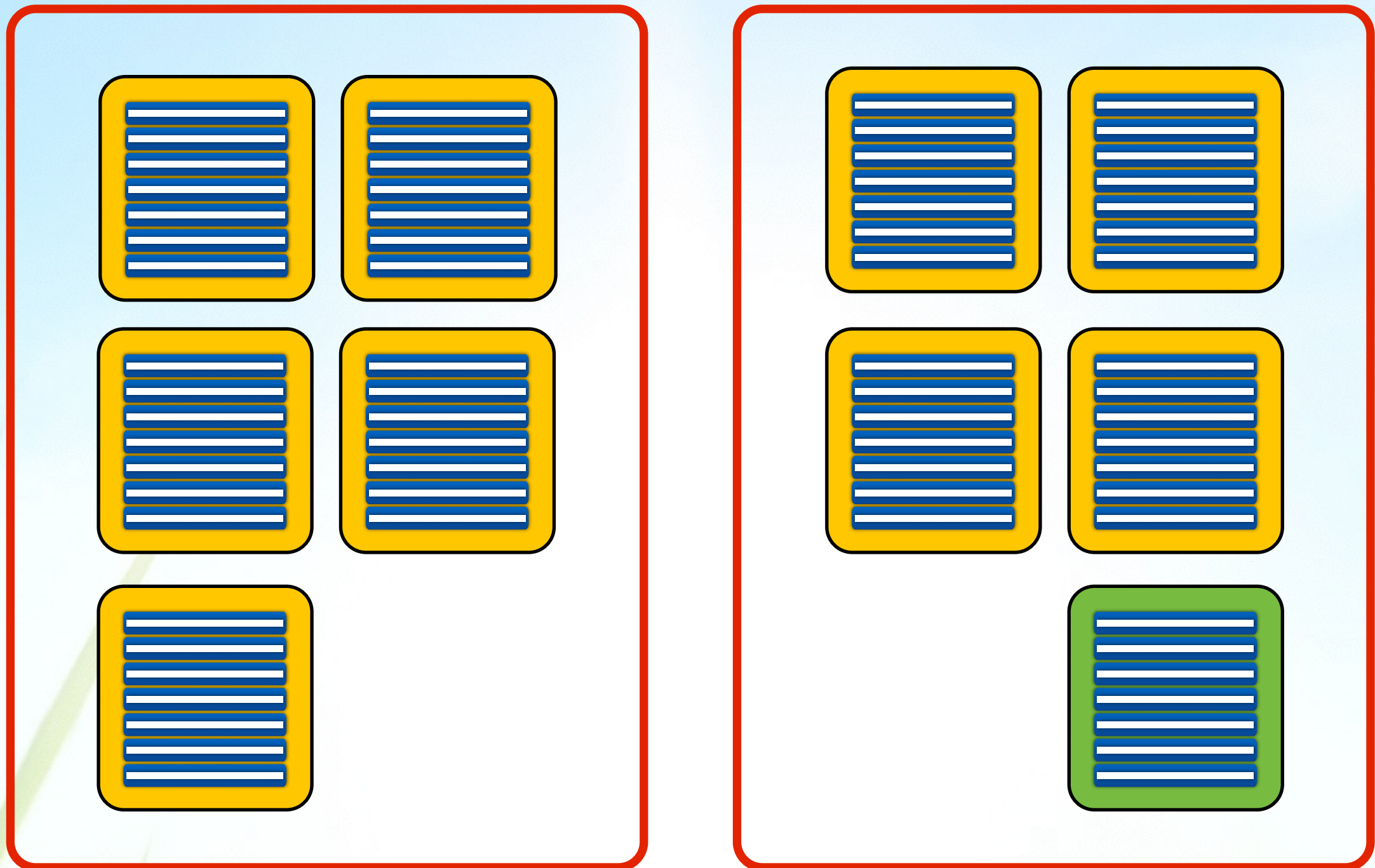




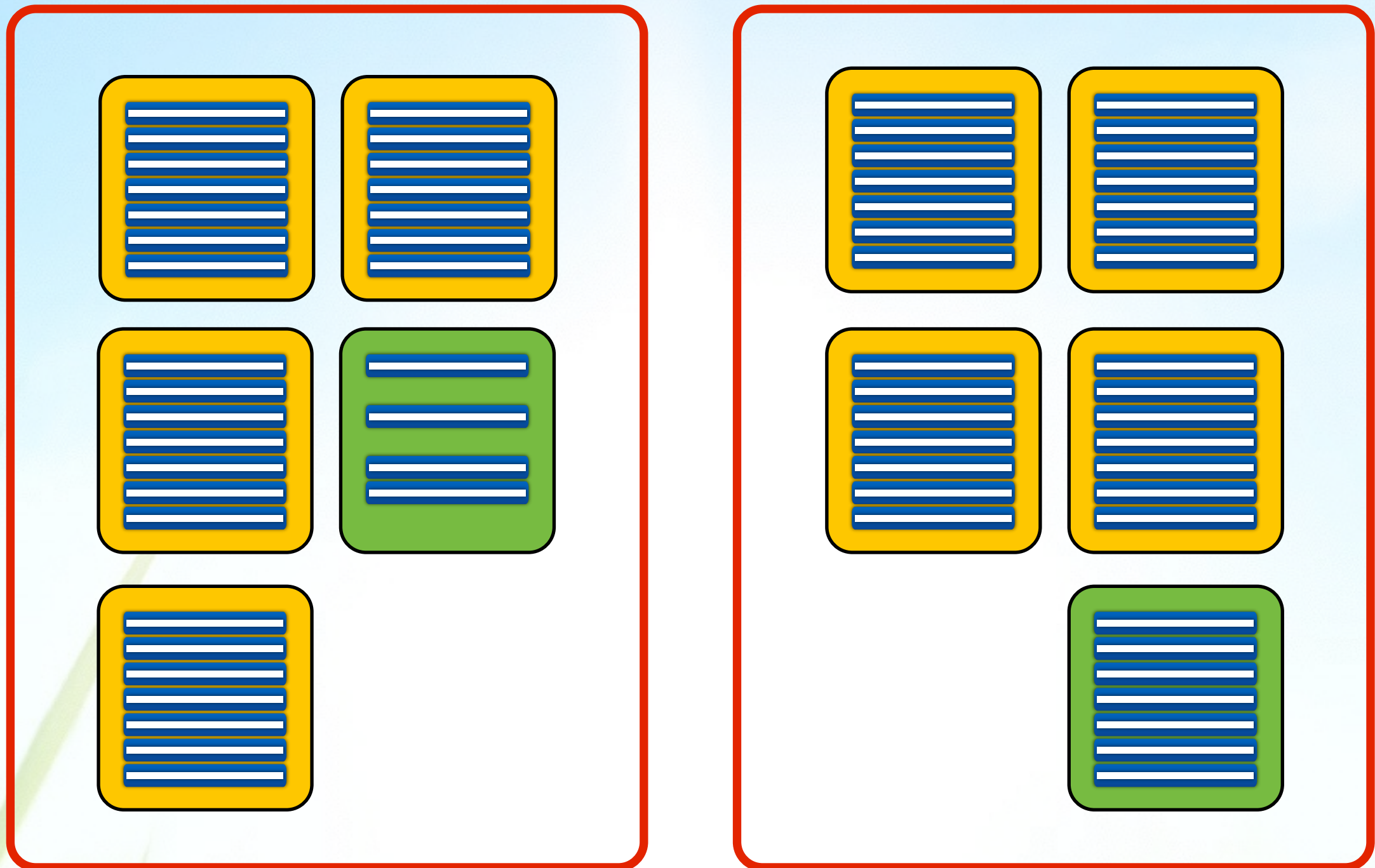
Crossover



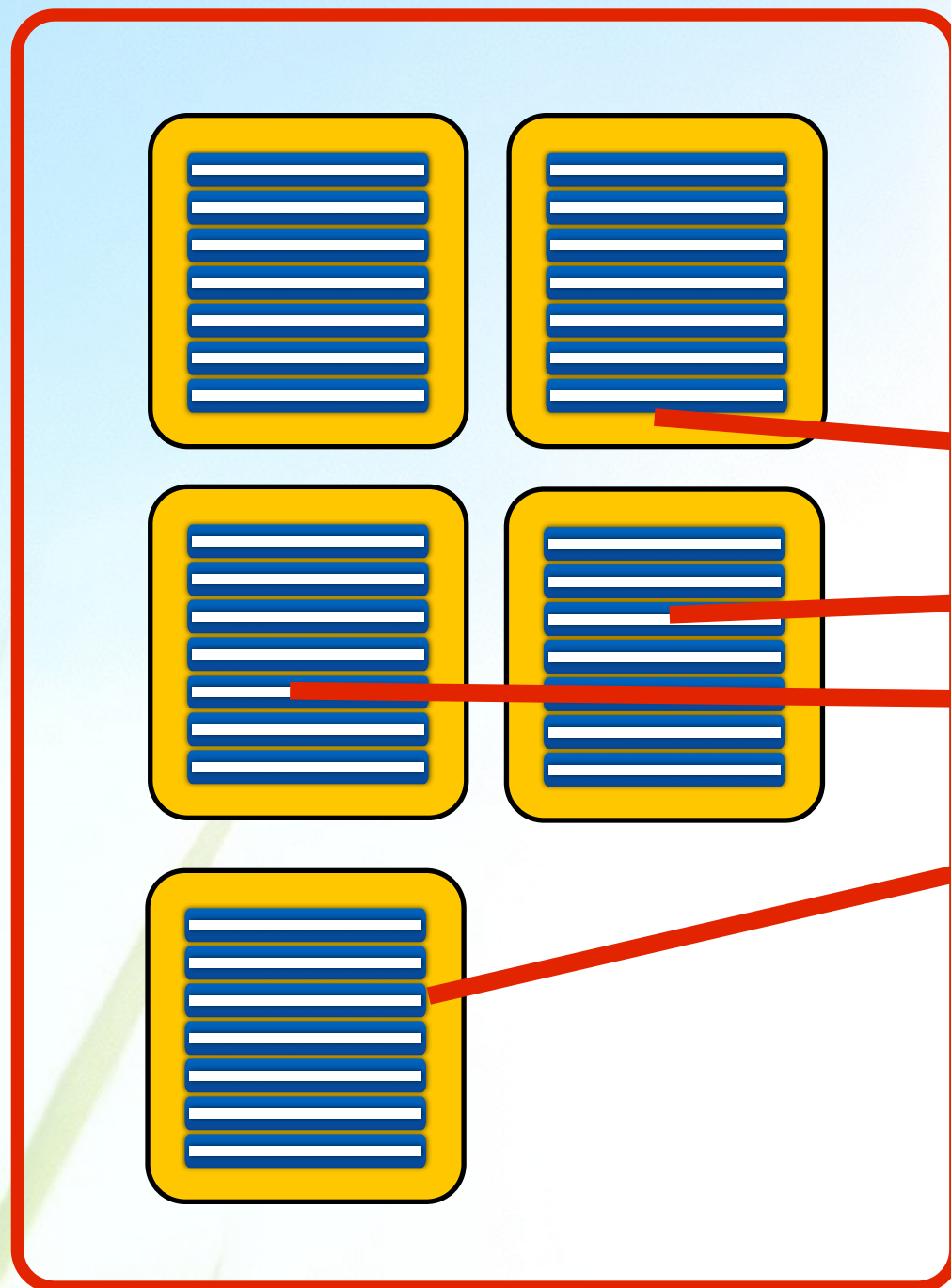
Mutation



Mutation



Fitness



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


Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

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 Sandbox

- Demo - Nondeterministic class
- Runtime library to execute tests

Testing multiple classes

Demo:

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

Experiment Exercise

- EvoSuite by default uses a combination of different coverage criteria.
- RQ1: Does the combination lead to larger test suites than just using branch coverage?
- RQ2: Does the combination lead to better test suites than just using branch coverage?

Setup

- Download and unzip:
http://evosuite.org/files/tutorial/Tutorial_Experiments.zip
- Maven project, but zip includes compiled code and dependencies
(`mvn compile dependency:copy-dependencies`)
- Where is EvoSuite?
`export EVOSUITE="java -jar /path/to/evosuite.jar"`

Treatments

- **Default criteria combination:**
`$EVOSUITE -class tutorial.Person`
- **Only branch coverage:**
`$EVOSUITE -class tutorial.Person -criterion branch`
- **Resulting data:**
`evosuite-report/statistics.csv`

Generating Data

http://evosuite.org/files/tutorial/Tutorial_Experiments.zip

- **Commons Options:**

- prefix tutorial -Dsearch_budget=20
 - Doutput_variables=configuration_id,TARGET_CLASS,Size,Length,MutationScore

- **Treatment 1:**

- \$EVOSUITE -Dconfiguration_id=Default <common options>

- **Treatment 2:**

- \$EVOSUITE -Dconfiguration_id=Branch -criterion branch <common options>

Cluster Experiments

- Demo - Sun Grid Engine

Analysing Data

- **R**
`analysis_scripts/analyse.R`
- **Python**
`analysis_scripts/analysis.py`
`analysis_scripts/plots.py`

Libs:

```
easy_install numpy  
easy_install matplotlib  
easy_install pandas  
easy_install scipy
```


Experiment Results

- RQ1: Does the combination lead to larger test suites than just using branch coverage?
- RQ2: Does the combination lead to better test suites than just using branch coverage?

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

Outline

- History
- Using EvoSuite
- **Extending EvoSuite**
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

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

Testing EvoSuite

Example Test:

```
public class NullStringSystemTest extends SystemTestBase {

    @Test
    public void testNullString() {
        EvoSuite evosuite = new EvoSuite();

        String targetClass = NullString.class.getCanonicalName();

        Properties.TARGET_CLASS = targetClass;

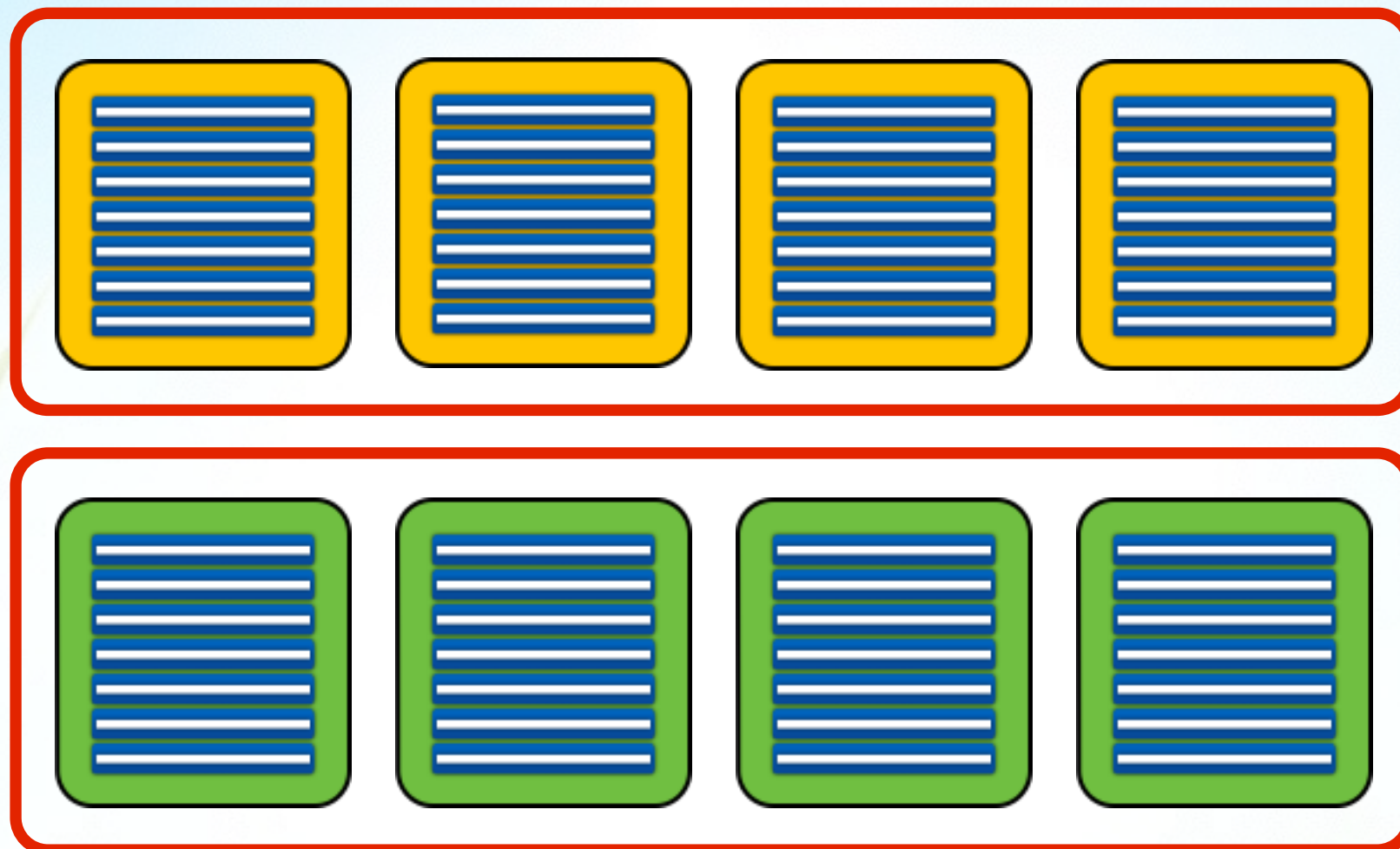
        String[] command = new String[] { "-generateSuite", "-class", targetClass };

        Object result = evosuite.parseCommandLine(command);
        GeneticAlgorithm<?> ga = getGAFromResult(result);
        TestSuiteChromosome best = (TestSuiteChromosome) ga.getBestIndividual();
        System.out.println("EvolvedTestSuite:\n" + best);

        int goals = TestGenerationStrategy.getFitnessFactories().
            get(0).getCoverageGoals().size(); // assuming single fitness function
        Assert.assertEquals("Wrong number of goals: ", 3, goals);
        Assert.assertEquals("Non-optimal coverage: ", 1d, best.getCoverage(), 0.001);
    }
}
```

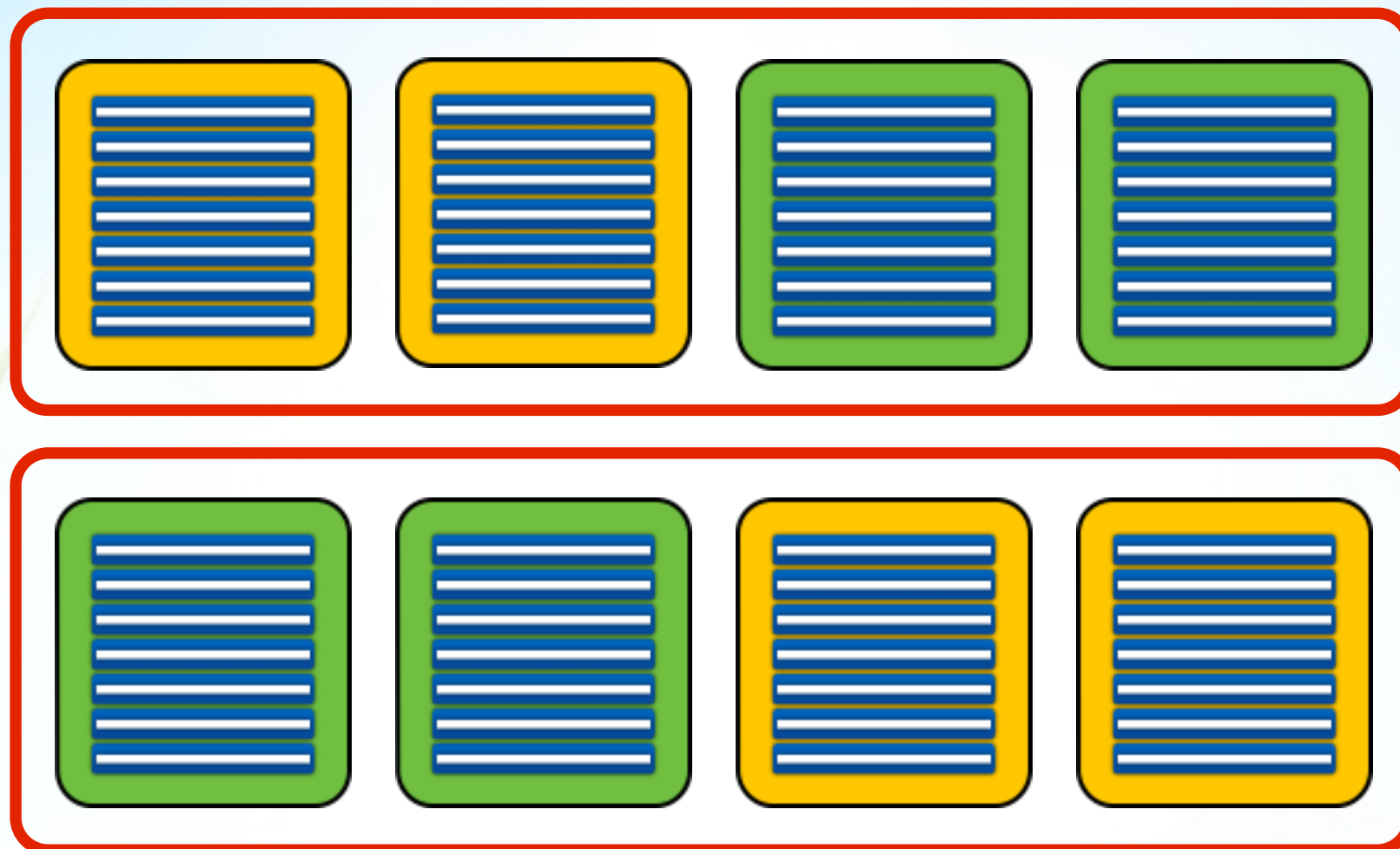

Extending EvoSuite

- (Artificial) Example: Middle point crossover



Extending EvoSuite

- (Artificial) Example: Middle point crossover



Extending EvoSuite

- (Artificial) Example: Middle point crossover

```
package org.evosuite.ga.operators.crossover;

import org.evosuite.ga.Chromosome;
import org.evosuite.ga.ConstructionFailedException;

public class MiddleCrossOver extends CrossOverFunction {
    @Override
    public void crossOver(Chromosome parent1, Chromosome parent2) throws ConstructionFailedException {
        // TODO
    }
}
```


Extending EvoSuite

- (Artificial) Example: Middle point crossover

```
@Test
public void testSinglePointCrossOver() throws ConstructionFailedException {
    DummyChromosome parent1 = new DummyChromosome(1, 2, 3, 4, 5, 6);
    DummyChromosome parent2 = new DummyChromosome(7, 8, 9, 10);

    MiddleCrossOver xover = new MiddleCrossOver();

    xover.crossOver(parent1, parent2);

    assertEquals(Arrays.asList(1, 2, 3, 9, 10), parent1.getGenes());
    assertEquals(Arrays.asList(7, 8, 4, 5, 6), parent2.getGenes());
}
```


Extending EvoSuite

- (Artificial) Example: Middle point crossover

```
@Override
public void crossOver(Chromosome parent1, Chromosome parent2)
    throws ConstructionFailedException {
    int middle1 = parent1.size()/2;
    int middle2 = parent2.size()/2;

    Chromosome t1 = parent1.clone();
    Chromosome t2 = parent2.clone();

    parent1.crossOver(t2, middle1, middle2);
    parent2.crossOver(t1, middle2, middle1);
}
```


Extending EvoSuite

- (Artificial) Example: Middle point crossover

```
public enum CrossoverFunction {  
    SINGLEPOINTRELATIVE, SINGLEPOINTFIXED, SINGLEPOINT, COVERAGE, MIDDLE  
}
```


Outline

- History
- Using EvoSuite
- **Extending EvoSuite**
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- **When to use and not to use EvoSuite**
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

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 whole test suite optimisation may not be the right choice

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, Gl, ...
- Sure, integrating EvoSuite should be easy

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- **When to use and not to use EvoSuite**
- Lessons learned building an SBST tool
- Things we are working on
- Ideas for future work

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- **Lessons learned building an SBST tool**
- Things we are working on
- Ideas for future work

I . Java

...is a weird language
and never ceases to surprise me

My personal enemy: Java Generics

Bytecode over sourcecode - yes!

2. Corner Cases

The more corner cases you cover

...the more can go wrong

...the more new corner cases you
will find

...the slower EvoSuite becomes

2. Corner Cases

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

3. Developers

...some really care only about coverage

...others don't care about coverage:

"I wouldn't normally in real life be aiming for 100% coverage. I'd probably end up with fewer tests without this tool but I couldn't tell you if they would be all the right tests."

...do not want their tests to be generated

...hate ugly tests

...don't like waiting

Talk to them!

3. Developers

```
public class Example {  
    private Example() {}  
  
    // ...  
}
```


4. Testing

Testing randomised algorithms is difficult

Make the implementation deterministic

Always use `LinkedHashSet` over `HashSet`,
`LinkedHashMap` over `HashMap`

Java reflection is not deterministic

Avoid static state (e.g. singletons)

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

5. Documentation

I don't comment my code

Students struggle

I spend more time explaining things than it would take me to implement them

6. Tool Comparisons

Reviewers want to see them

I don't like doing them

It's impossible to make them fair

Contact tool authors

Report bugs

Make your own tools *usable*

7. Open Source

“The source code will be released under an open source library (most likely GPL2) at a later point, as soon as a number of refactorings are completed.” — FSE’11 tool paper appendix

Public GitHub repo: 2015

It will never be clean enough, just release it!

8. Licensing

License matters

Google will not touch GPL

BSD, MIT - do you want others to become rich with your idea?

Gnu Lesser Public License, Apache

9. Tool Papers

The first one will be cited

The rest no one will cite

It shouldn't be this way

10. Robustness

Creating a robust tool...

...is a huge effort

...it will never be finished

EvoSuite is a black hole swallowing all my time!

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- **Lessons learned building an SBST tool**
- Things we are working on
- Ideas for future work

Outline

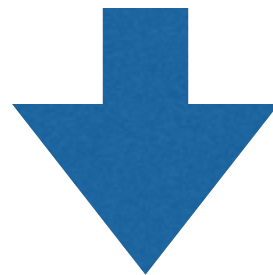
- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- **Things we are working on**
- Ideas for future work

Stuff we are working on...

- Increasing coverage...
- Readability optimisation
- Better environment handling
- Mocking and private reflection
- Finding out how developers benefit most from using test generation
- User studies, replications

Method Names

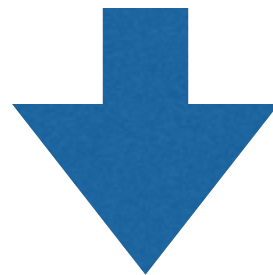
```
@Test(timeout = 4000)
public void test3() throws Throwable {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```



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


Variable Names

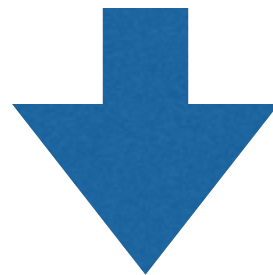
```
@Test(timeout = 4000)
public void testFooReturningFalse() throws Throwable {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```



```
@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);  
}
```


Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- **Things we are working on**
- Ideas for future work

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- **Ideas for future work**

I. 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?

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

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- **Ideas for future work**

Outline

- History
- Using EvoSuite
- Extending EvoSuite
- When to use and not to use EvoSuite
- Lessons learned building an SBST tool
- Things we are working on
- 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/>