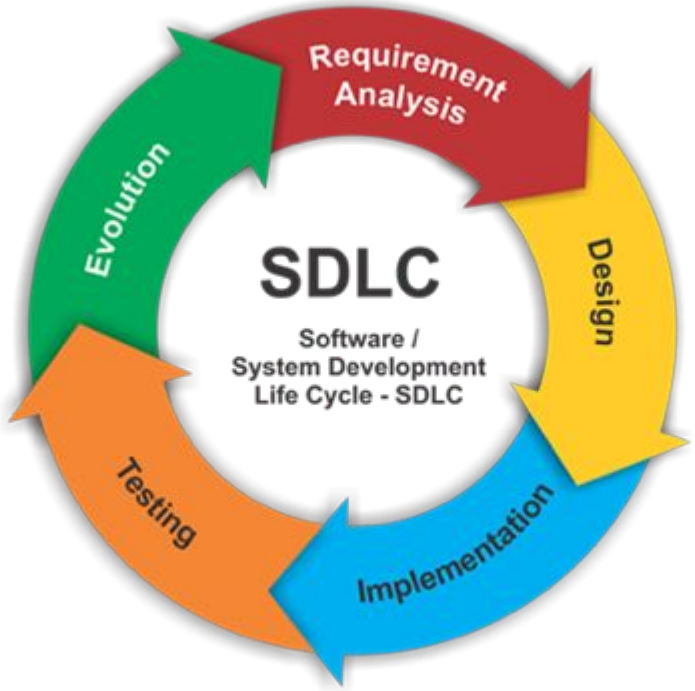
EvoSuite: Automatic test suite generation for object-oriented software

Authors: Gordon Fraser and Andrea Arcuri

Presenter: Xiaolong(Marlon) Ma

Life cycle of software development



Different level of testing

- Unit test
- Integration test
- End-to-end test
- Performance test
- Security test
- ect. etc.

What is automation test?

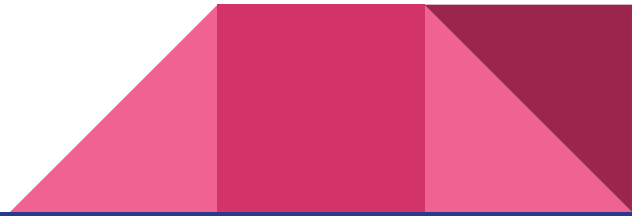
- **Automation testing** is a Software **testing** technique to **test** and compare the actual outcome with the expected outcome.
- This can be achieved by writing **test** scripts or using any **automation testing** tool.
- **Test automation** is used to **automate** repetitive tasks and other **testing** tasks which are difficult to perform manually

Test and Development video link

https://www.youtube.com/watch?v=zhxcrcF1w10&list=PLzL47ht5IgCFIIk3cqyJ0cr_9I6QmM6wb&index=9&t=0s

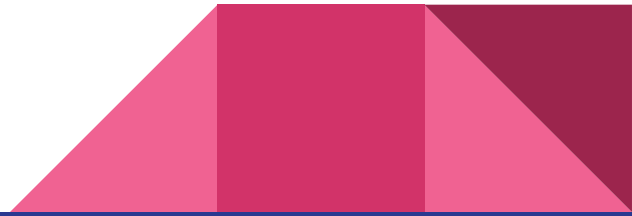
Motivations

- Faster Feedback.
- Accelerated Results.
- Reduced Business Expenses.
- Testing Efficiency Improvement
- Higher Overall Test Coverage.
- Reusability of Automated Tests.
- Earlier Detection of Defects.
- Thoroughness in Testing.
- Faster Time-to-Market

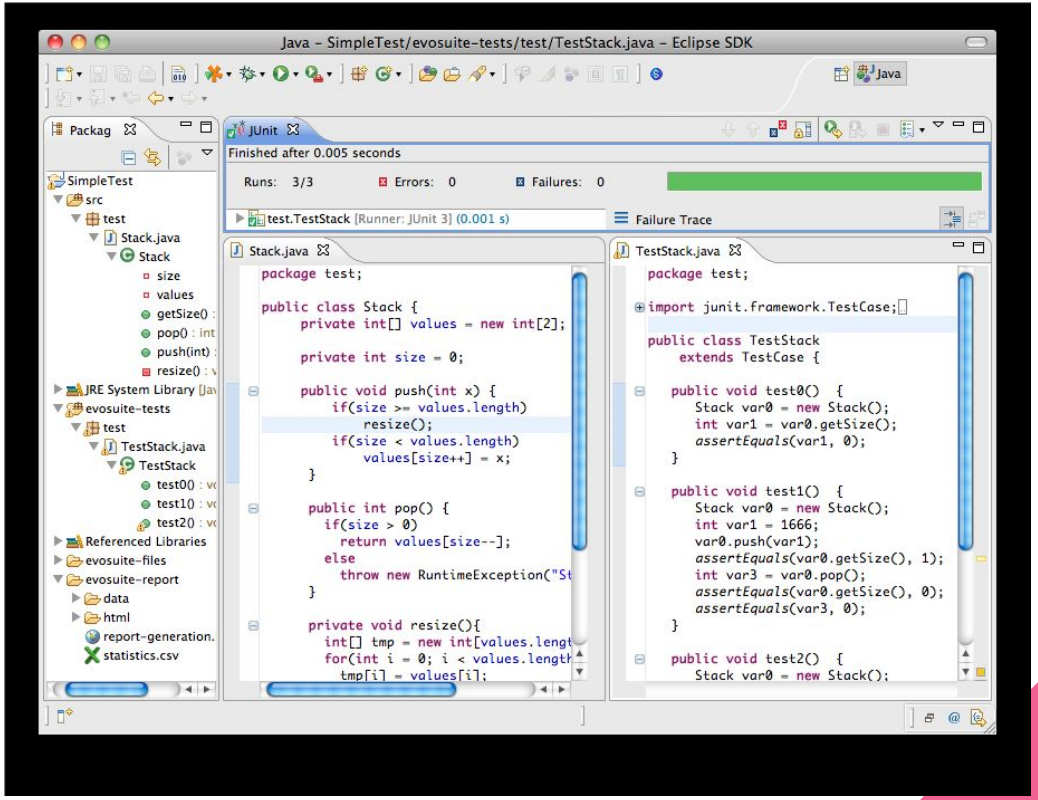


EvoSuite

- EvoSuite is a tool that automatically generates test cases with assertions for classes written in Java code.
- EvoSuite applies a novel hybrid approach that generates and optimizes whole test suites towards satisfying a coverage criterion.
- EvoSuite suggests possible oracles by adding small and effective sets of assertions that concisely summarize the current behavior



EvoSuite


Java – SimpleTest/evosuite-tests/test/TestStack.java – Eclipse SDK
Java
Packag
JUnit
Finished after 0.005 seconds
Runs: 3/3
Errors: 0
Failures: 0
test.TestStack [Runner: JUnit 3] (0.001 s)
Failure Trace
SimpleTest
src
test
Stack.java
Stack
size
values
getSize() :
pop() : int
push(int) :
resize() :
JRE System Library [Jav
evosuite-tests
test
TestStack.java
TestStack
test0() : vo
test1() : vo
test2() : vo
Referenced Libraries
evosuite-files
evosuite-report
data
html
report-generation.
statistics.csv
Stack.java
package test;
public class Stack {
private int[] values = new int[2];
private int size = 0;
public void push(int x) {
if(size >= values.length)
resize();
if(size < values.length)
values[size++] = x;
}
public int pop() {
if(size > 0)
return values[size--];
else
throw new RuntimeException("St
}
private void resize(){
int[] tmp = new int[values.lengt
for(int i = 0; i < values.length
tmp[i] = values[i];
TestStack.java
package test;
import junit.framework.TestCase;
public class TestStack
extends TestCase {
public void test0() {
Stack var0 = new Stack();
int var1 = var0.getSize();
assertEquals(var1, 0);
}
public void test1() {
Stack var0 = new Stack();
int var1 = 1666;
var0.push(var1);
assertEquals(var0.getSize(), 1);
int var3 = var0.pop();
assertEquals(var0.getSize(), 0);
assertEquals(var3, 0);
}
public void test2() {
Stack var0 = new Stack();

Problem definition

- Set of classes under test(CUTs)
- Automatically generate test cases for CUT
- Maximize coverage(statement, mutation, etc.)
- Generate assertions: capture behavior
- Small test suites(engineers will look at them)

Methodology

- Random Testing
- Seeding
- Optimisation problem
- Heuristics
- Hill climbing
- Genetic algorithms

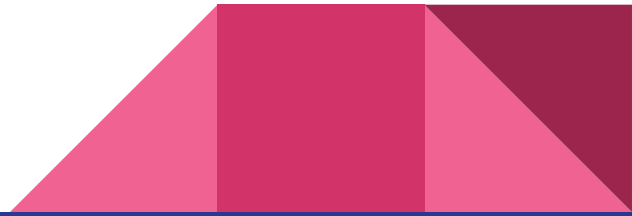
Fitness function

- Evaluate how good a test is.
- Does it cover new instructions
- How close is the test to cover new instructions(optimization problem)

HC Example

```
if(x*2 == 42)
    return true
return false
```

- Starting with random x = 15
- d(“x*2 == 42”, 15) = |15*2 - 42| = 12
- Neighbourhood +- 1: 14 and 16
- d(14) = 14, d(16) = 10 -> moving to x = 16
- Repeat for x = 16, i.e +- 1
  d(15) = 12, d(17) = 8 -> moving to x = 17
- Repeat till the solution x = 21



GA Example

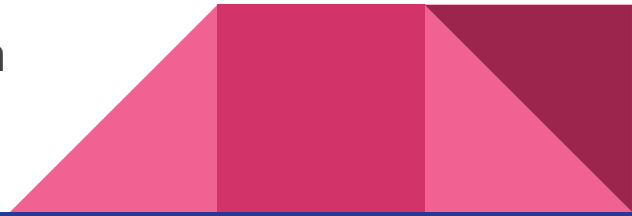
- Keep a population of solutions(eg 50)
- Calculate fitness of each solution in population
- Mate solutions based on fitness

  Eg. Create offspring by recombining some parts of parents representation

- Kill off the worst solution

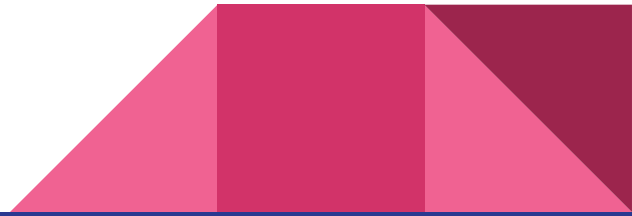
  E.g. keep same population size

- Evolve several generations until find optimal solution



Tests depend on environment

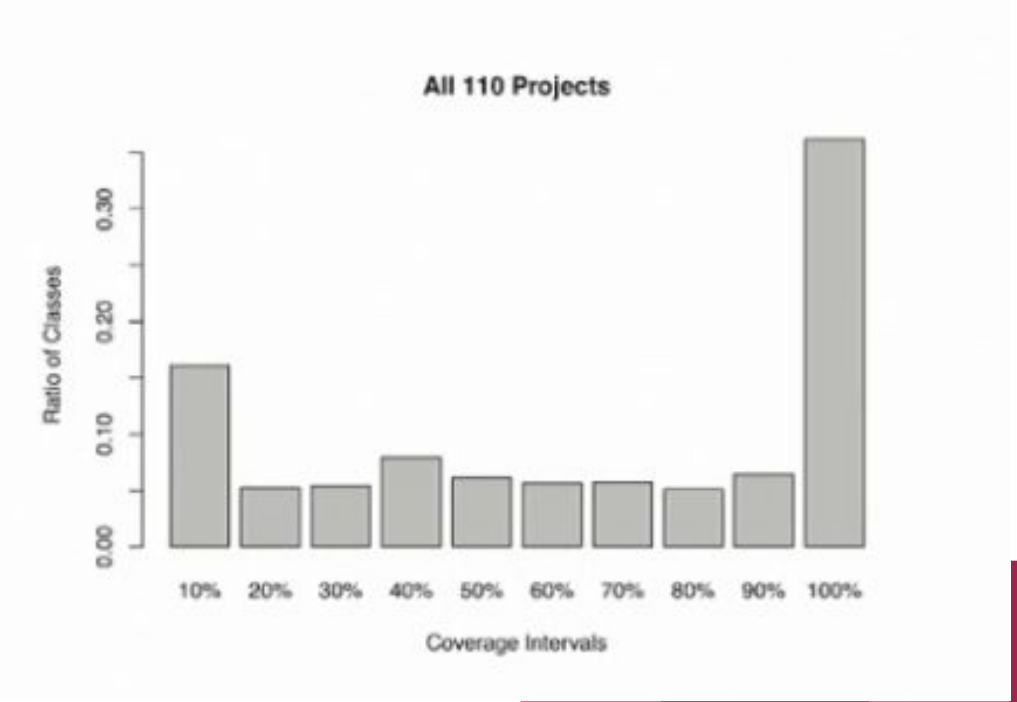
1. Detect if the CUTs interacts with the environment
2. Simulate environment inputs



Evaluation

- > 30,000 classes

Open source: avg 71%



Code example

```
public class Foo {
    public static double divide(int x, int y){
        if(y==0) throw new  IllegalArgumentException();
       return x / y;
    }
}
```

Code example

```
public static void main(String[] args)
  throws Exception{

  Random rand = new Random();
    for(int i=0; i<10; i++){
        int x = rand.nextInt(), y = rand.nextInt();
        double res = Foo.divide(x,y);
        System.out.println(
           "@Test public void test"+i+"(){");
        System.out.println(
           "  assertEquals(“+res
            +", Foo.divide("+x+", "+y+");}");
    }
}
```

Example of automatic code generation

https://www.youtube.com/watch?v=drPJmvZZ0gs&list=PLzL47ht5IgCFIIk3cqyJ0cr_9I6QmM6wb&index=8&t=343s

Conclusion

- It is possible to efficiently generate tests automatically
- Cannot totally replace human manual testing

Questions

1. What is the life cycle of software development
2. Please list three different level of testing
3. List three motivations of automation testing