



UNIVERSITY OF  
**LEICESTER**

Department  
Of  
Informatics

# EvoSuite: Generación Automática de Casos de Prueba

José Miguel Rojas



*Some slides borrowed from Prof. G. Fraser*

UCM, Madrid, España

# CV

- Ingeniería Informática  
Universidad Autónoma Gabriel René Moreno, Bolivia (2001–2006)
- Doctorado en Software y Sistemas  
Universidad Politécnica de Madrid, España (2009–2013)
  - Tesis Doctoral: Generación Automática de Casos de Prueba en Programación Orientada a Objetos
- Investigador Post-doctoral (Research Associate)  
The University of Sheffield, Reino Unido (2014–2017)
- Profesor Asistente (Lecturer)  
University of Leicester, Reino Unido (2017–presente)

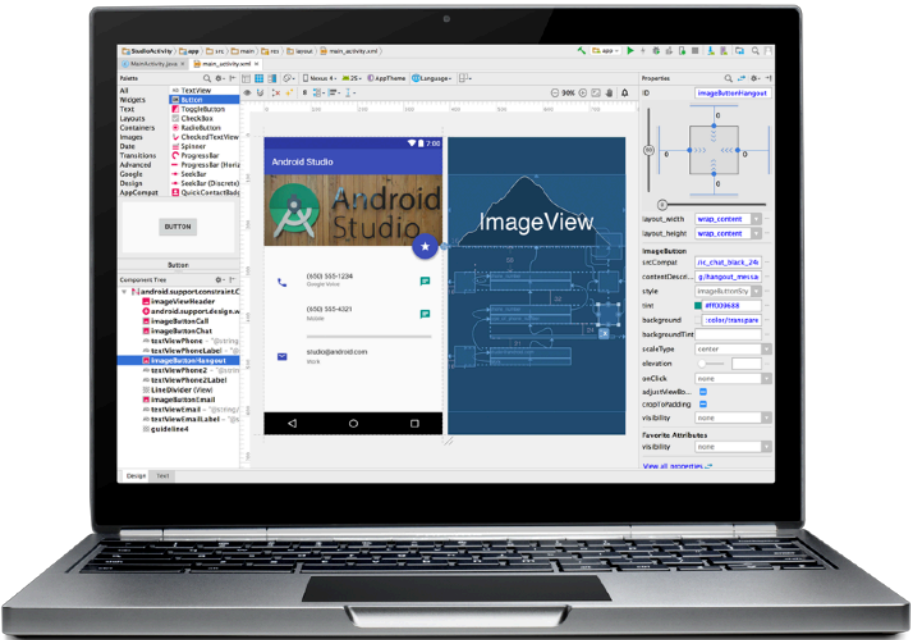


# Disclaimer

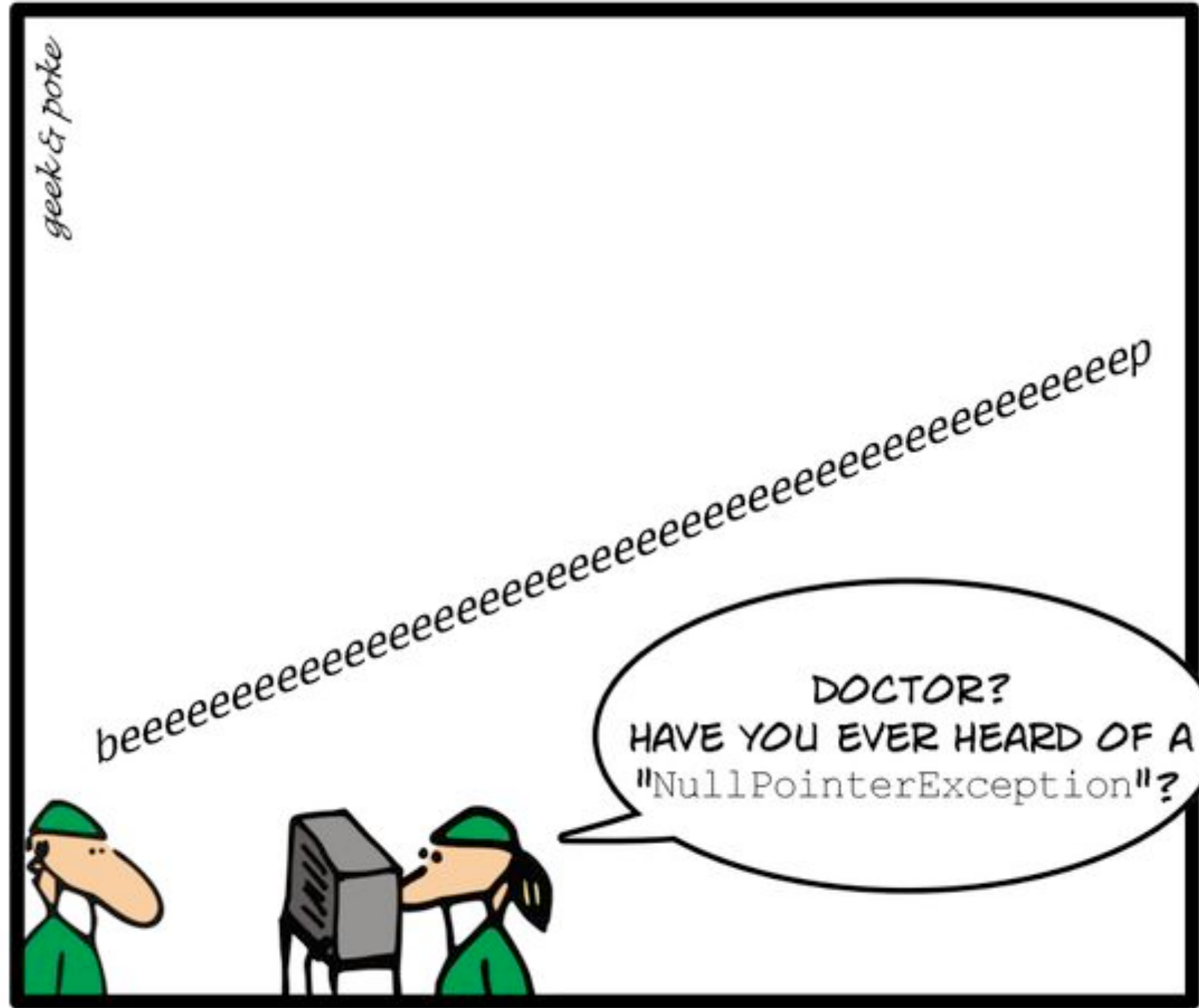








# Bugs



## RECENTLY IN THE OPERATING ROOM



## Windows

A fatal exception 0E has occurred at 0028:C00068F8 in UxD UMM(01) + 000059F8. The current application will be terminated.

- \* Press any key to terminate the application.
- \* Press CTRL+ALT+DEL to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue





Ariane 5, primer vuelo, 1996

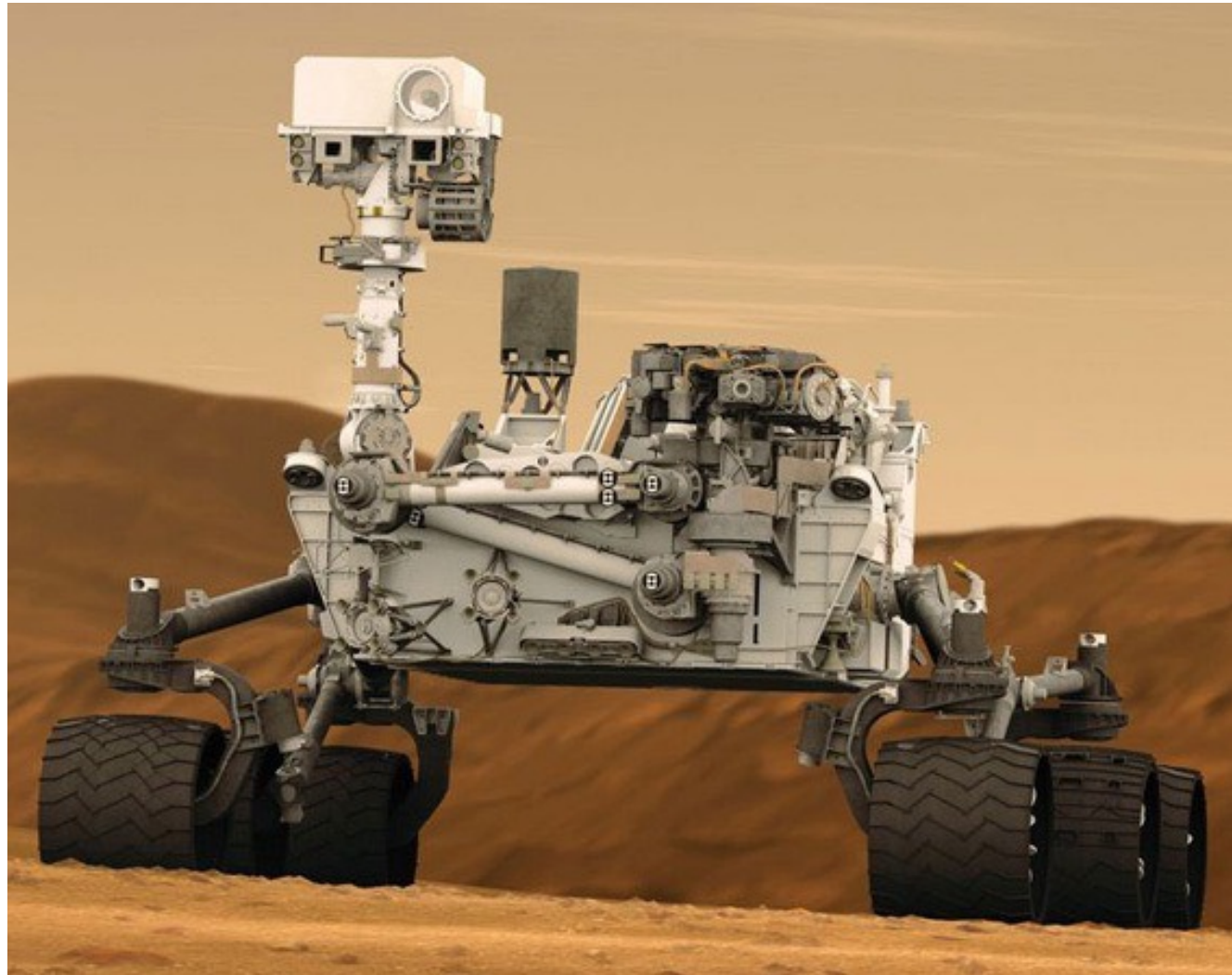




Therac-25, 1985-87

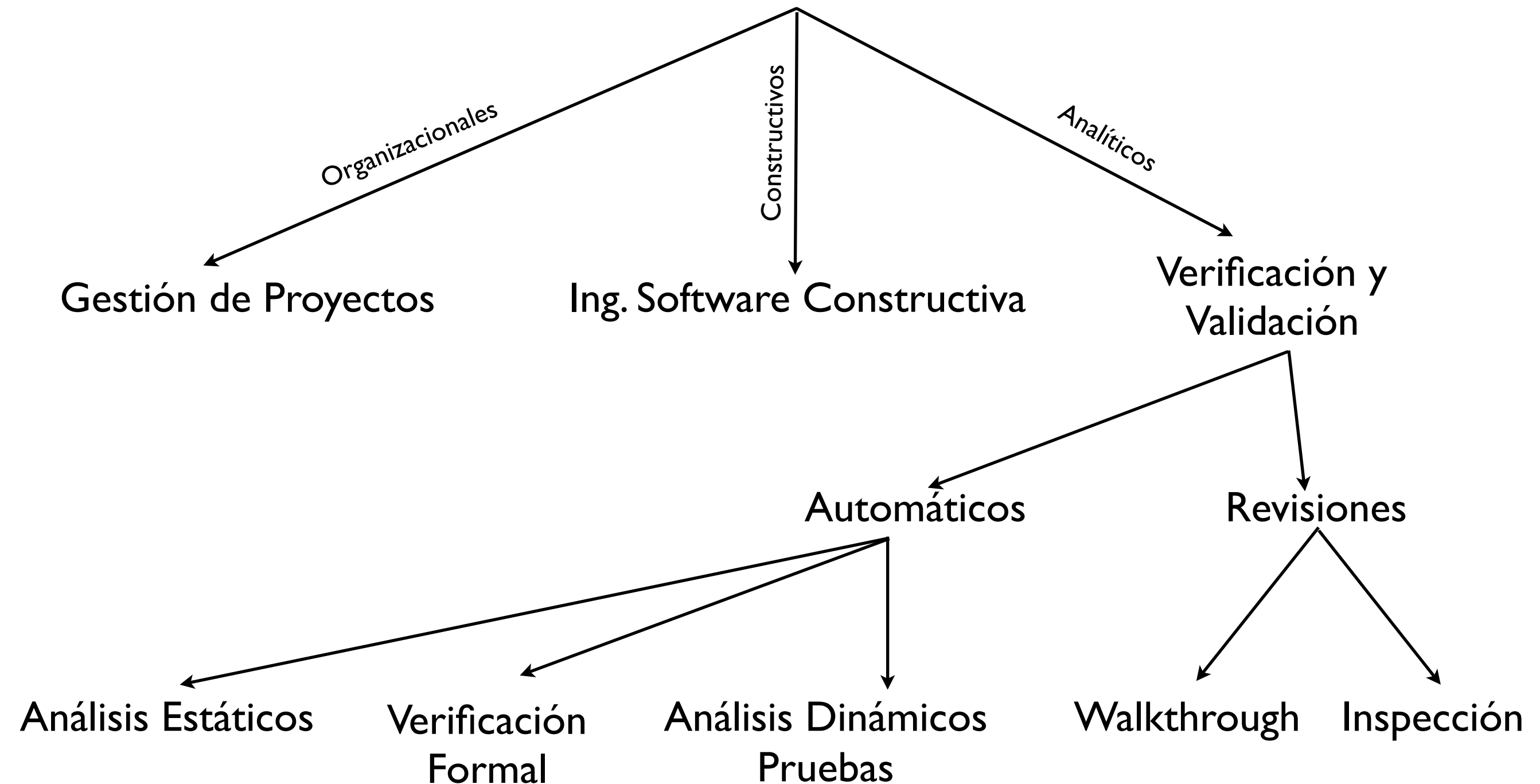


# ¿Software libre de errores?

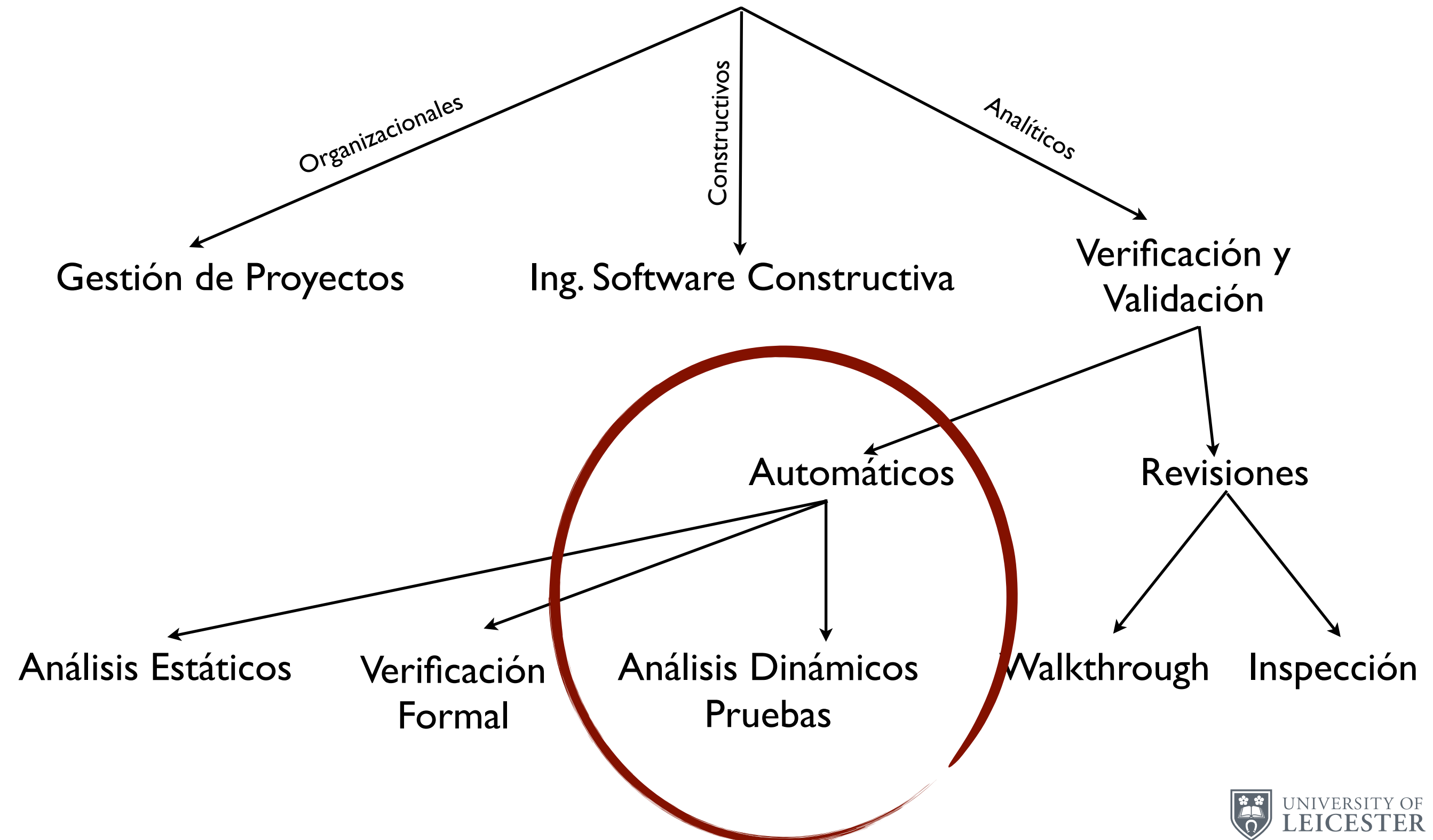




# Métodos para la Calidad del Software



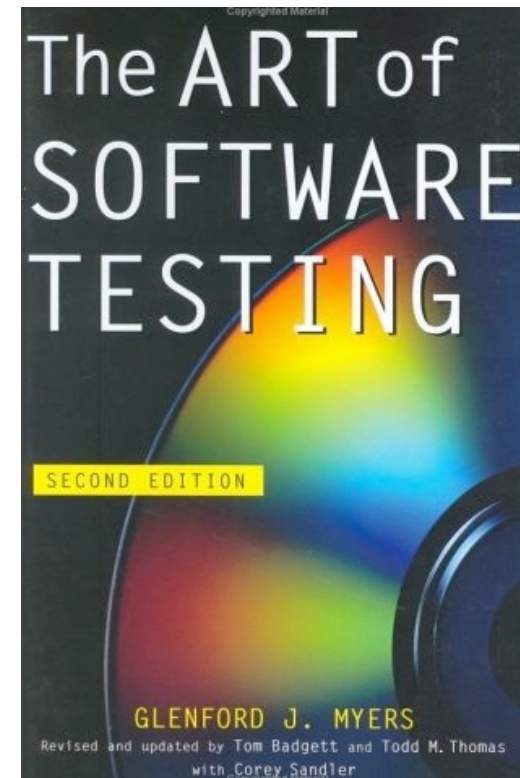
# Métodos para la Calidad del Software





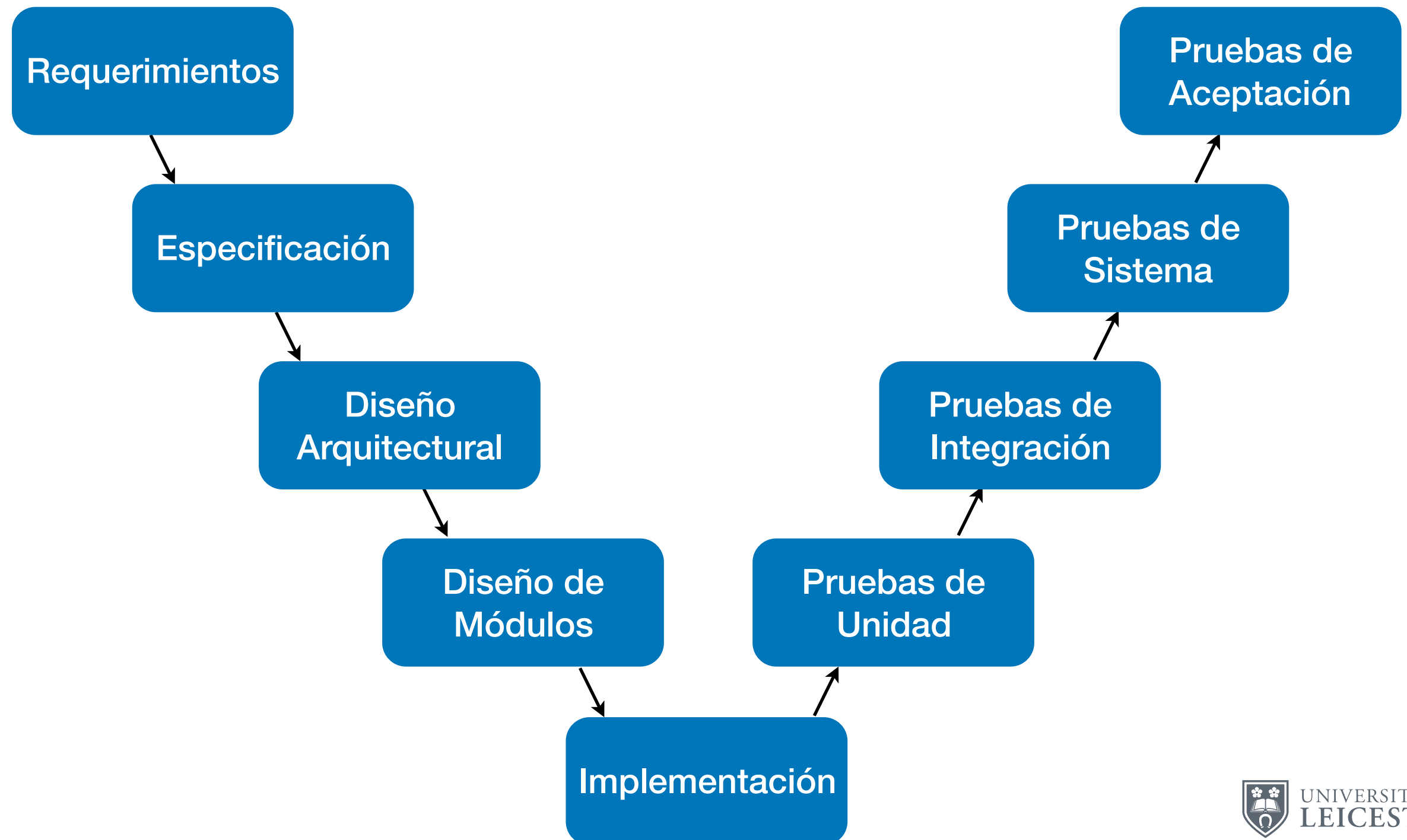
# Pruebas de Software

“La selección de entradas para un sistema con la intención de provocar fallos en dicho sistema, revelando de tal forma la presencia de errores”

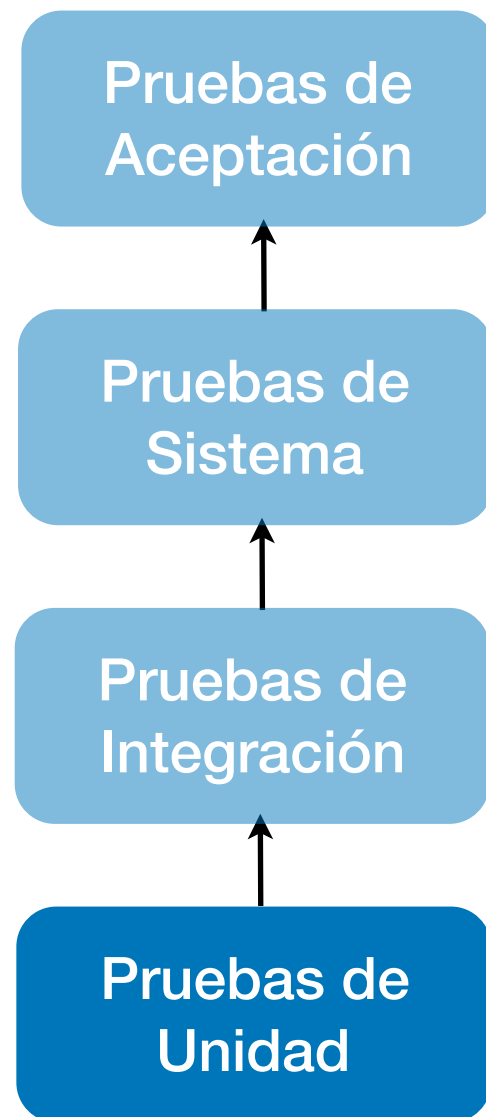


“Las pruebas de software solo pueden demostrar la presencia de errores, no su ausencia.” (E. Dijkstra)

# Pruebas de Software



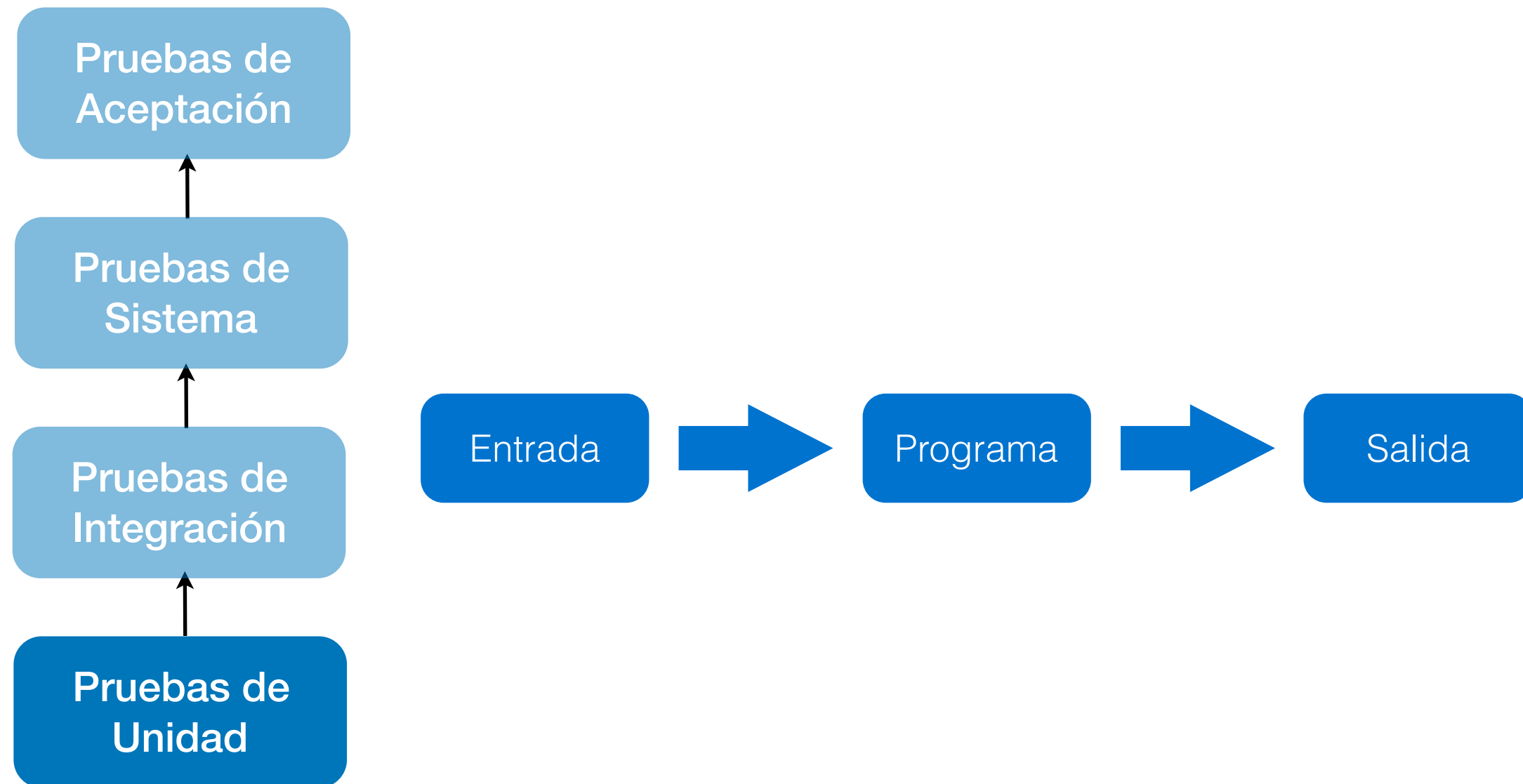
# Pruebas de Unidad



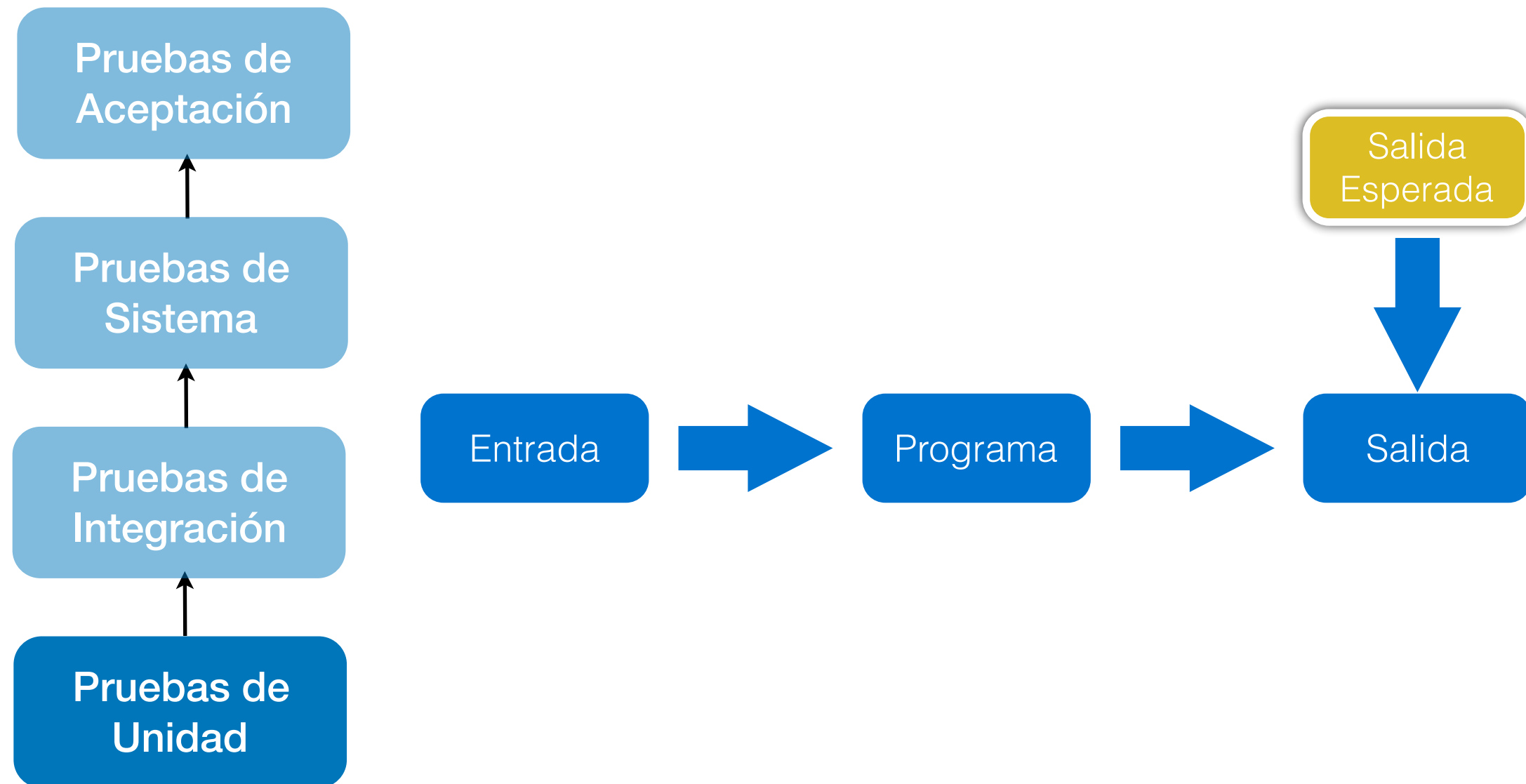
1. Ejecutar programa, usando datos de prueba (*input*)
2. Verificar salida del programa (*output*), usando oráculo de pruebas (*test oracle*)



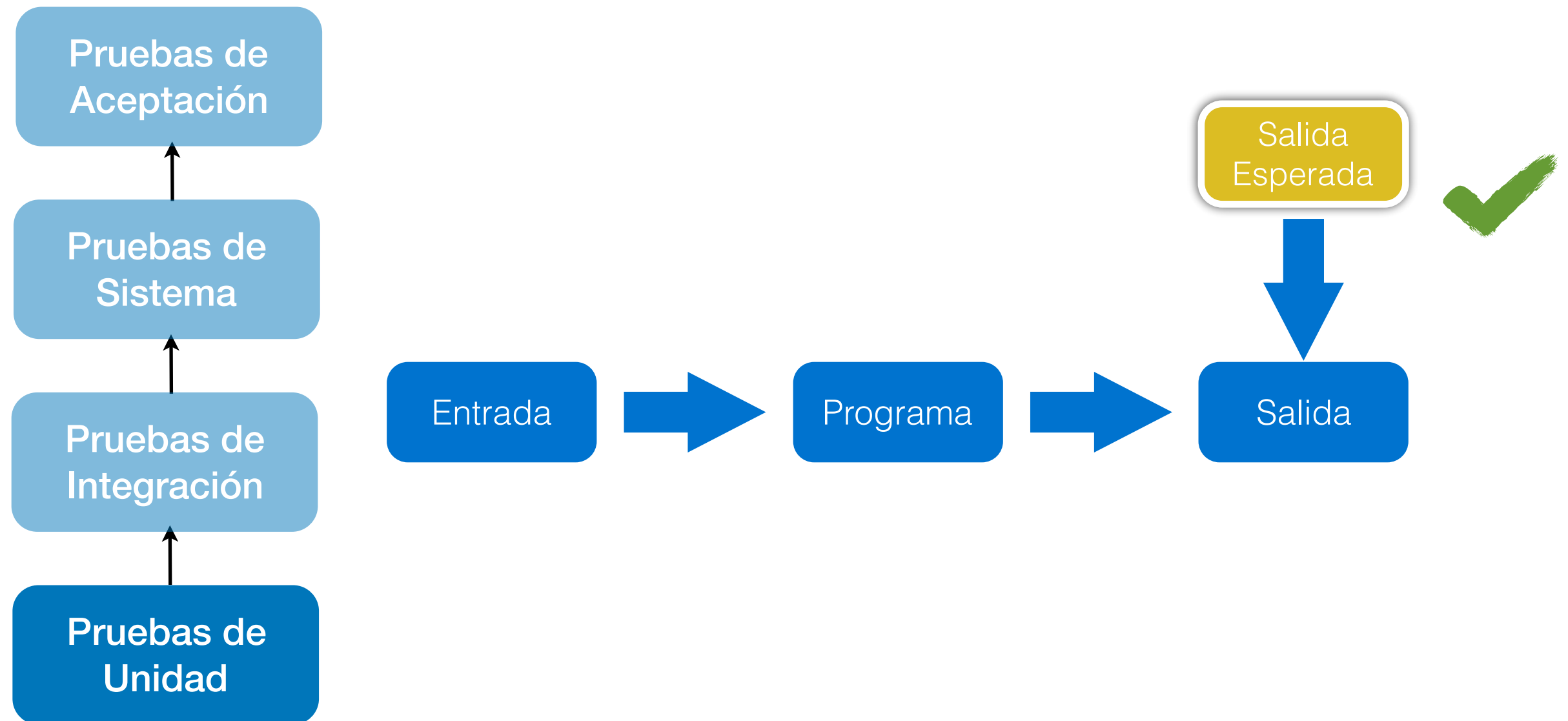
# Pruebas de Unidad



# Pruebas de Unidad

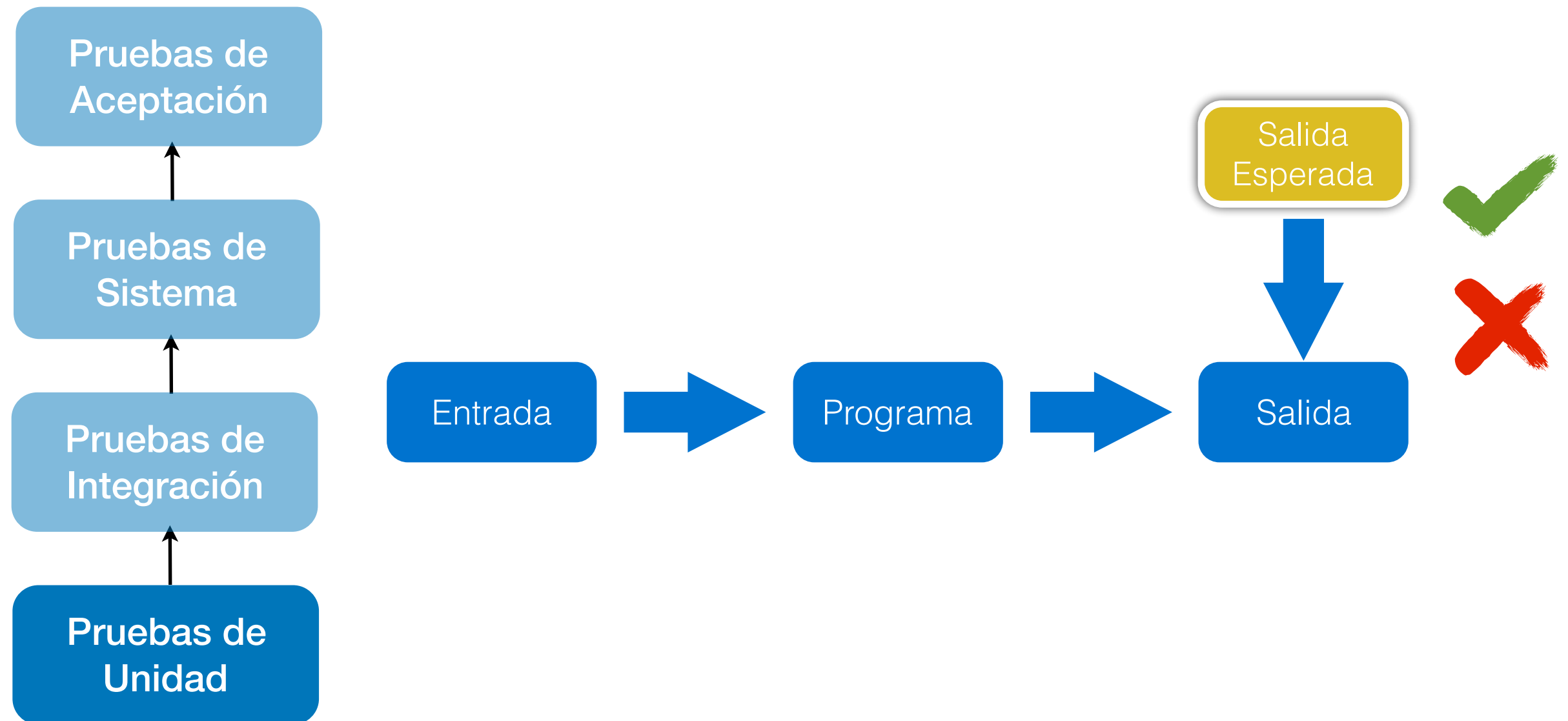


# Pruebas de Unidad

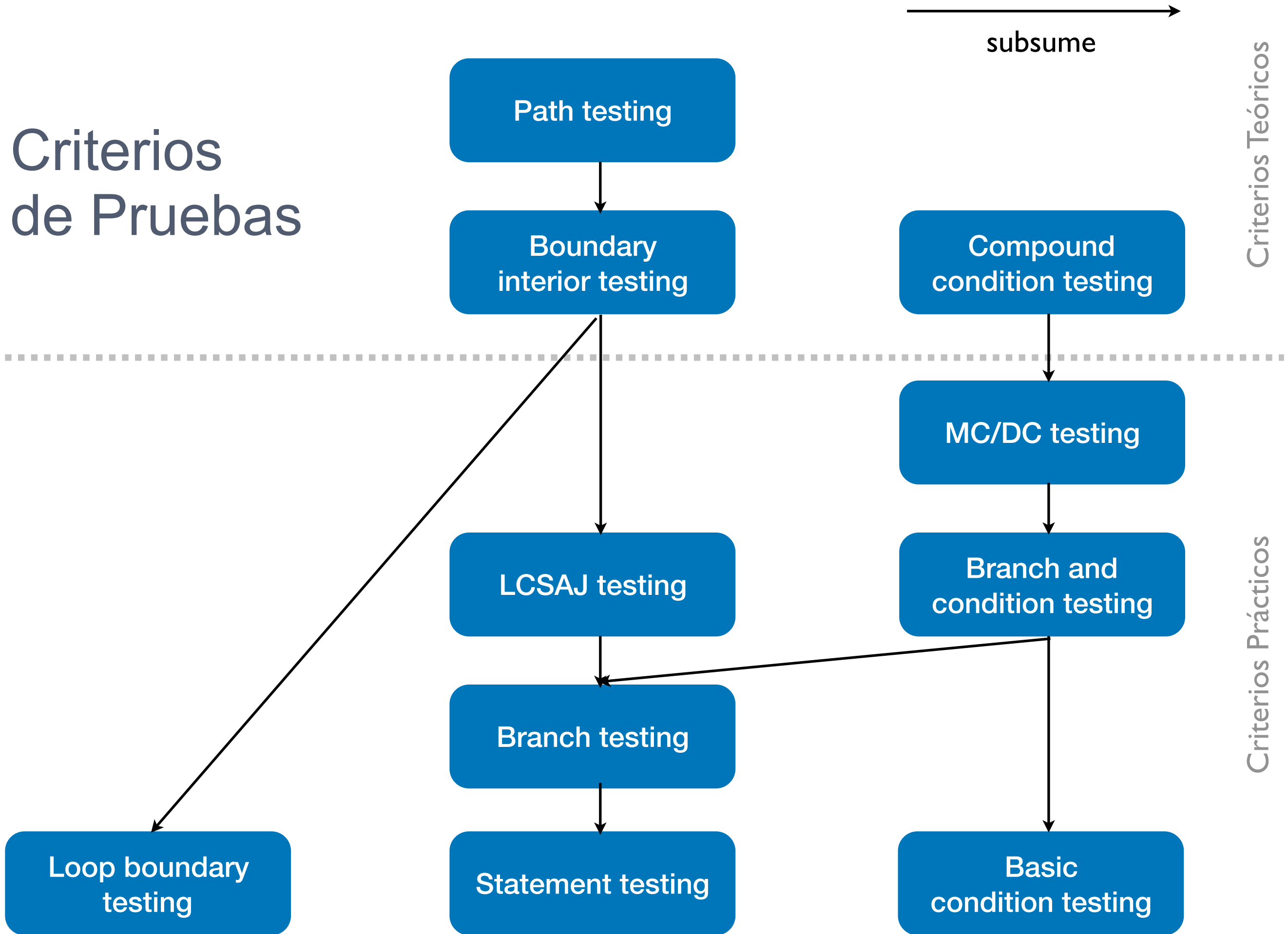




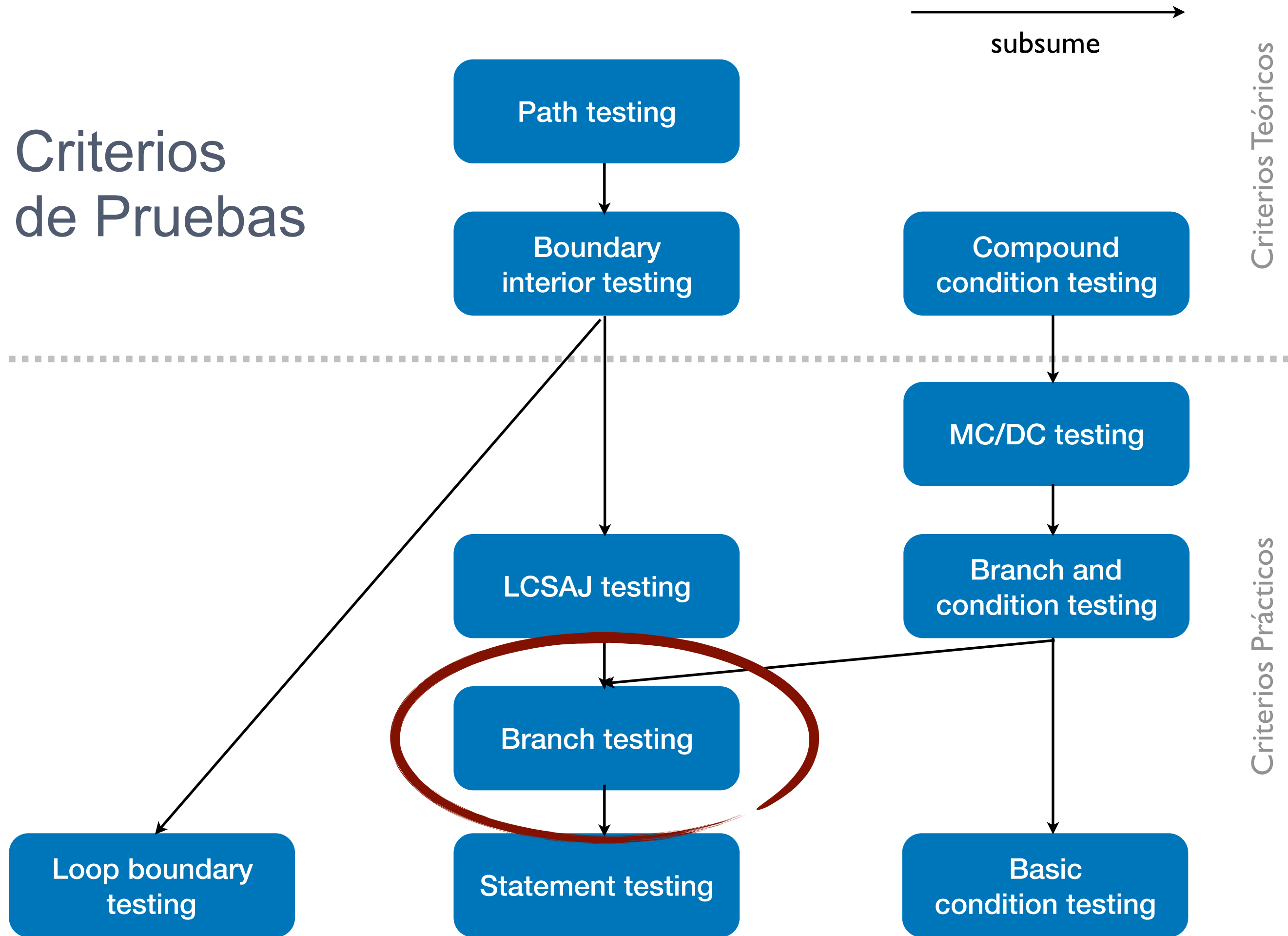
# Pruebas de Unidad

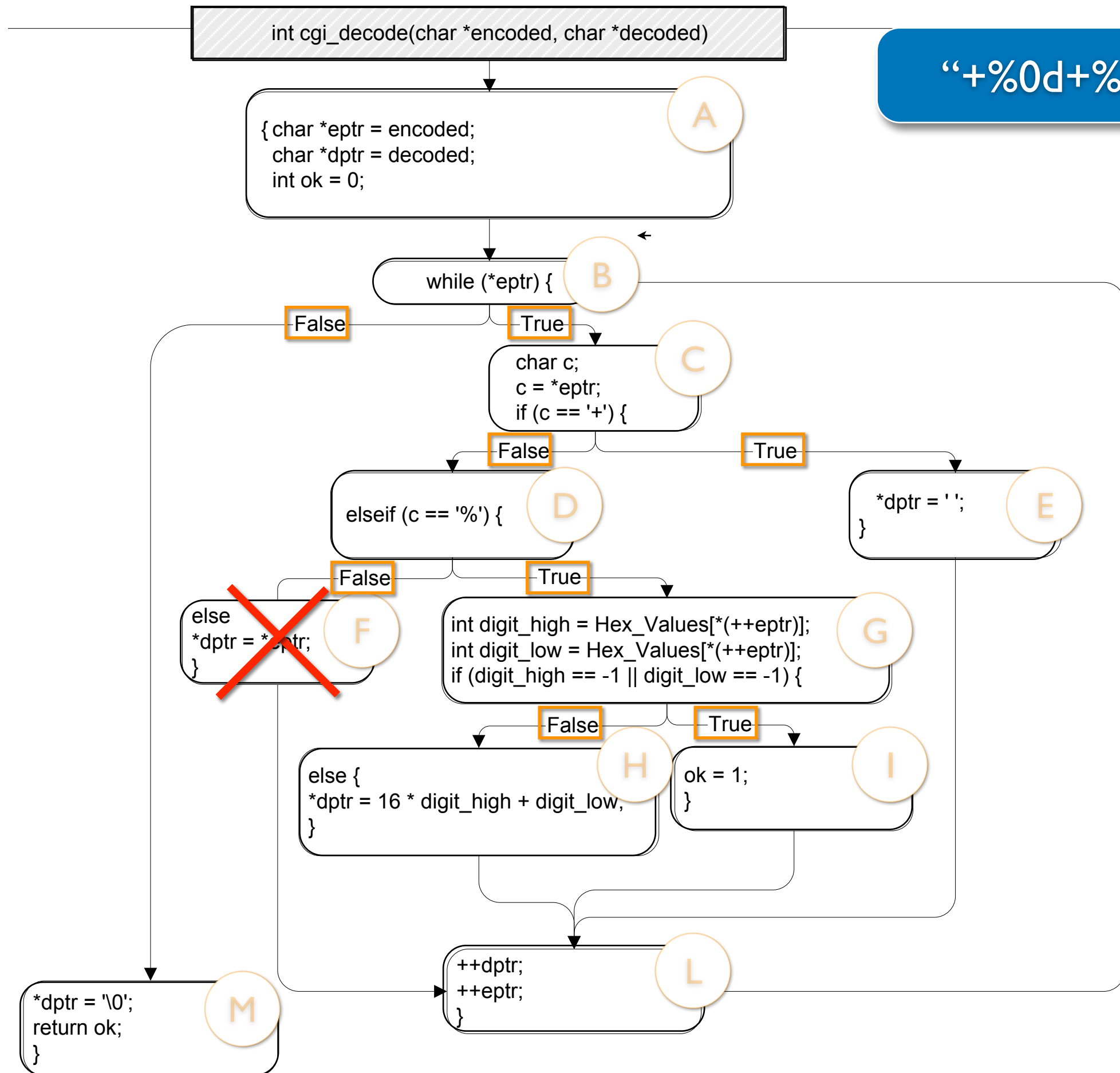


# Criterios de Pruebas



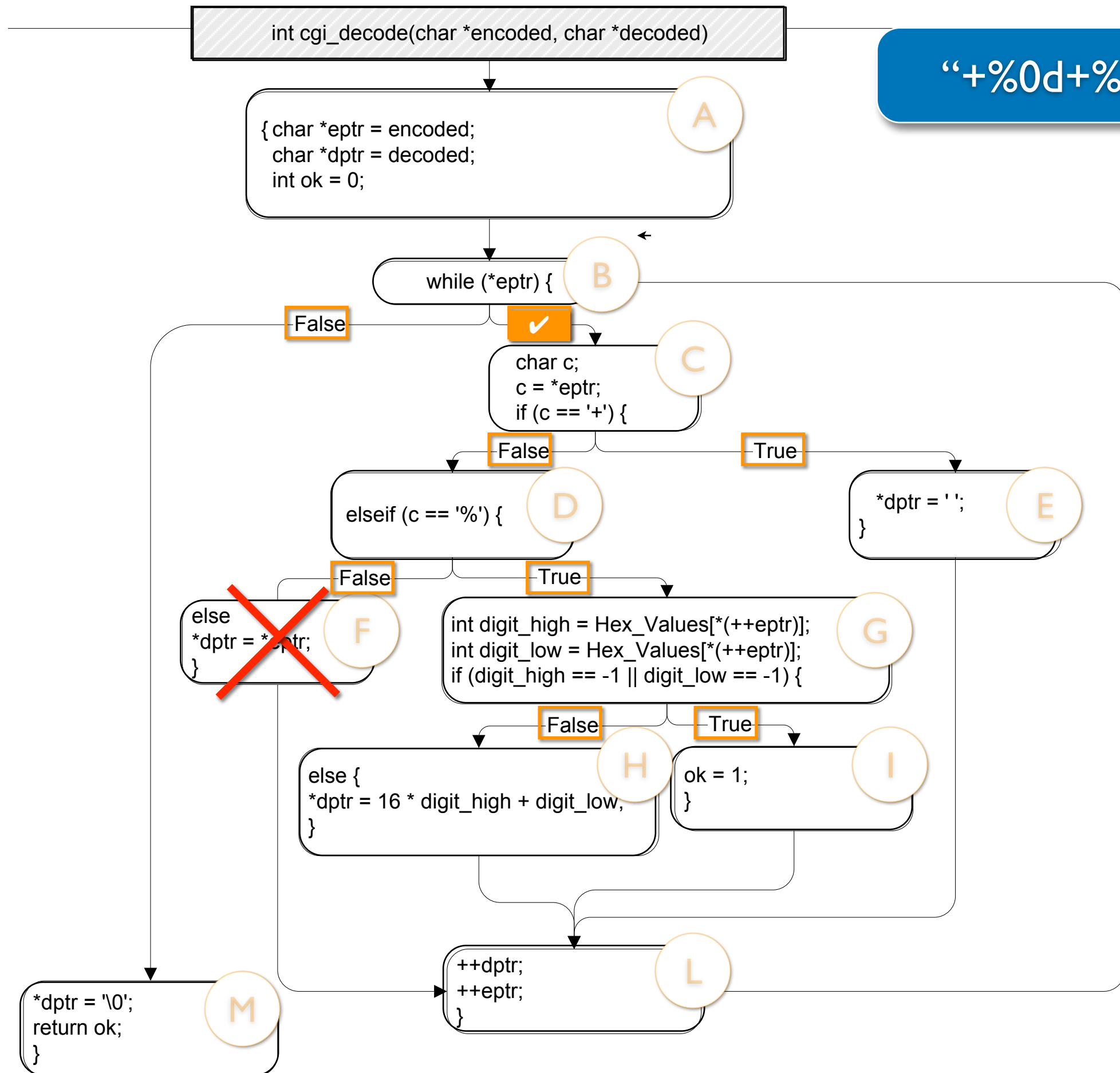
# Criterios de Pruebas



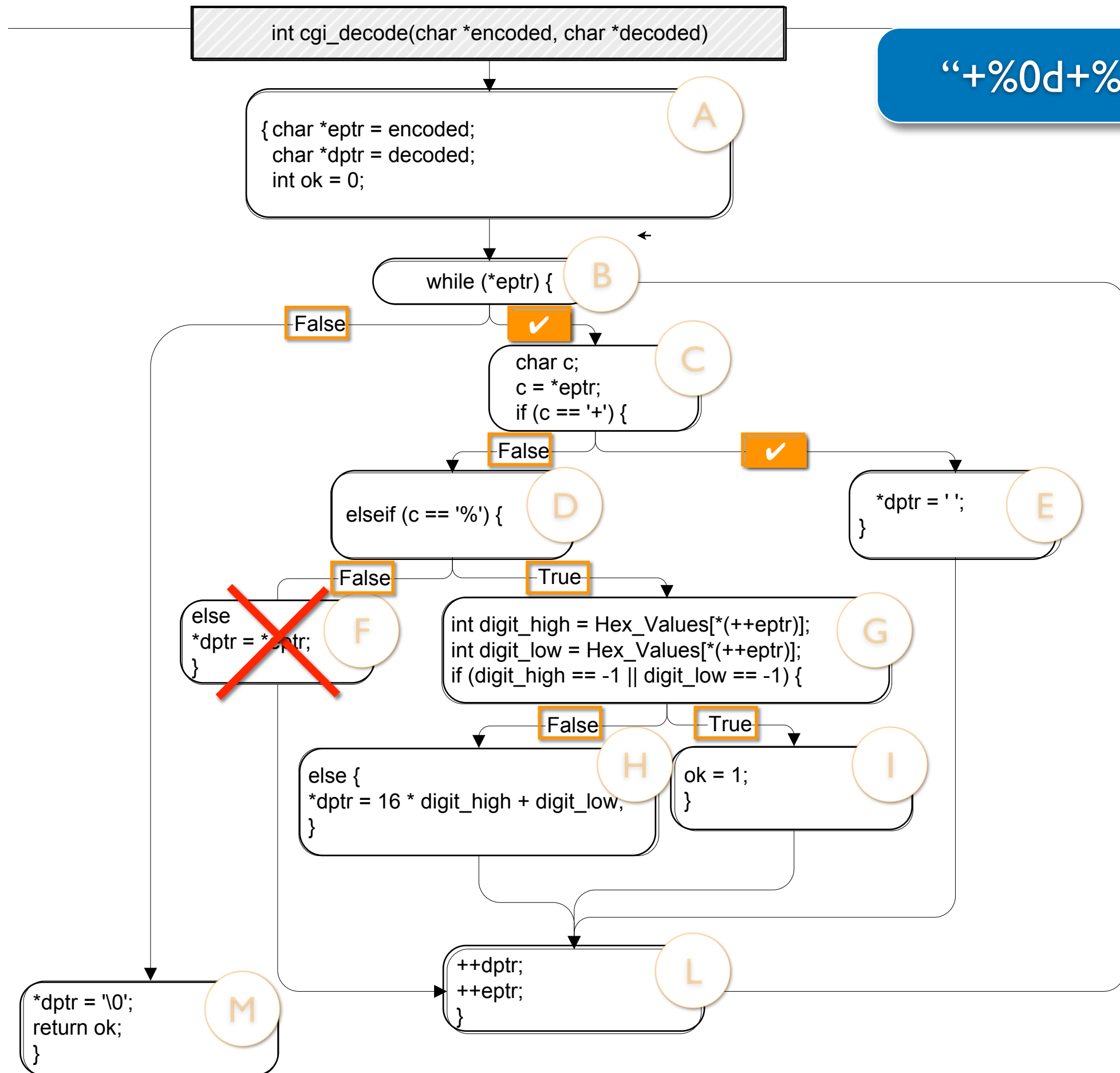


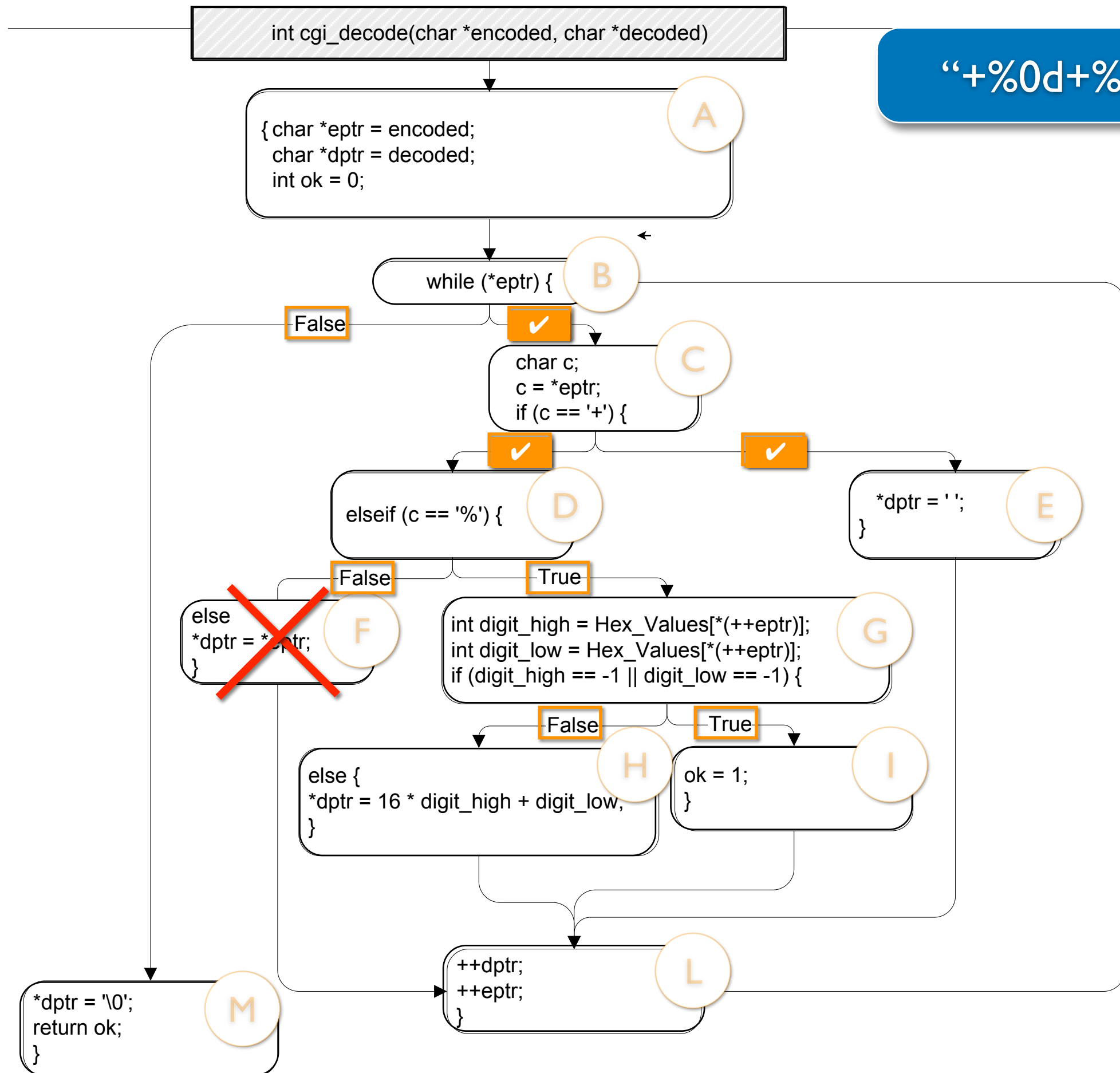
“+ %0d + %4j”



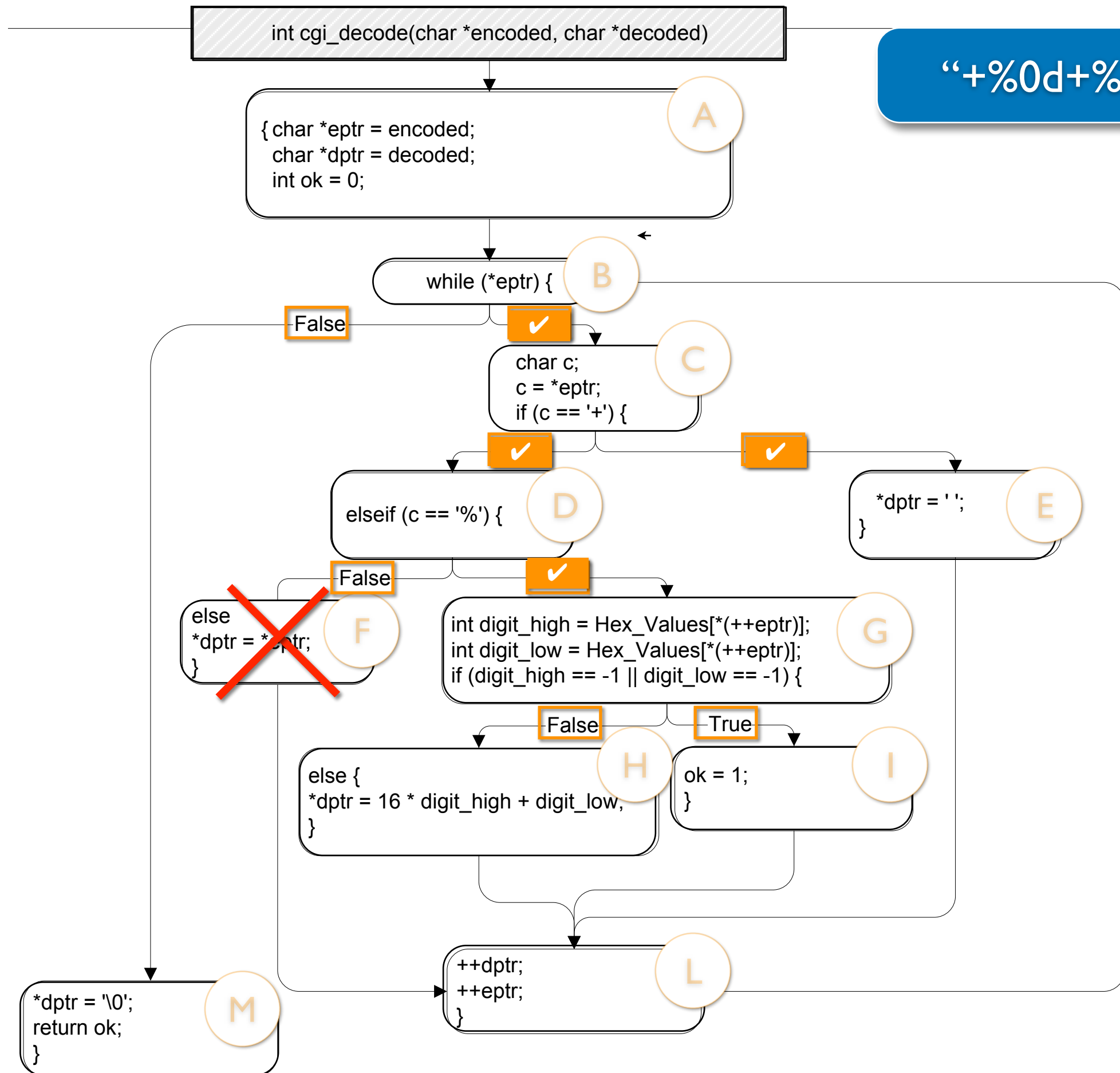


“+ %0d + %4j”

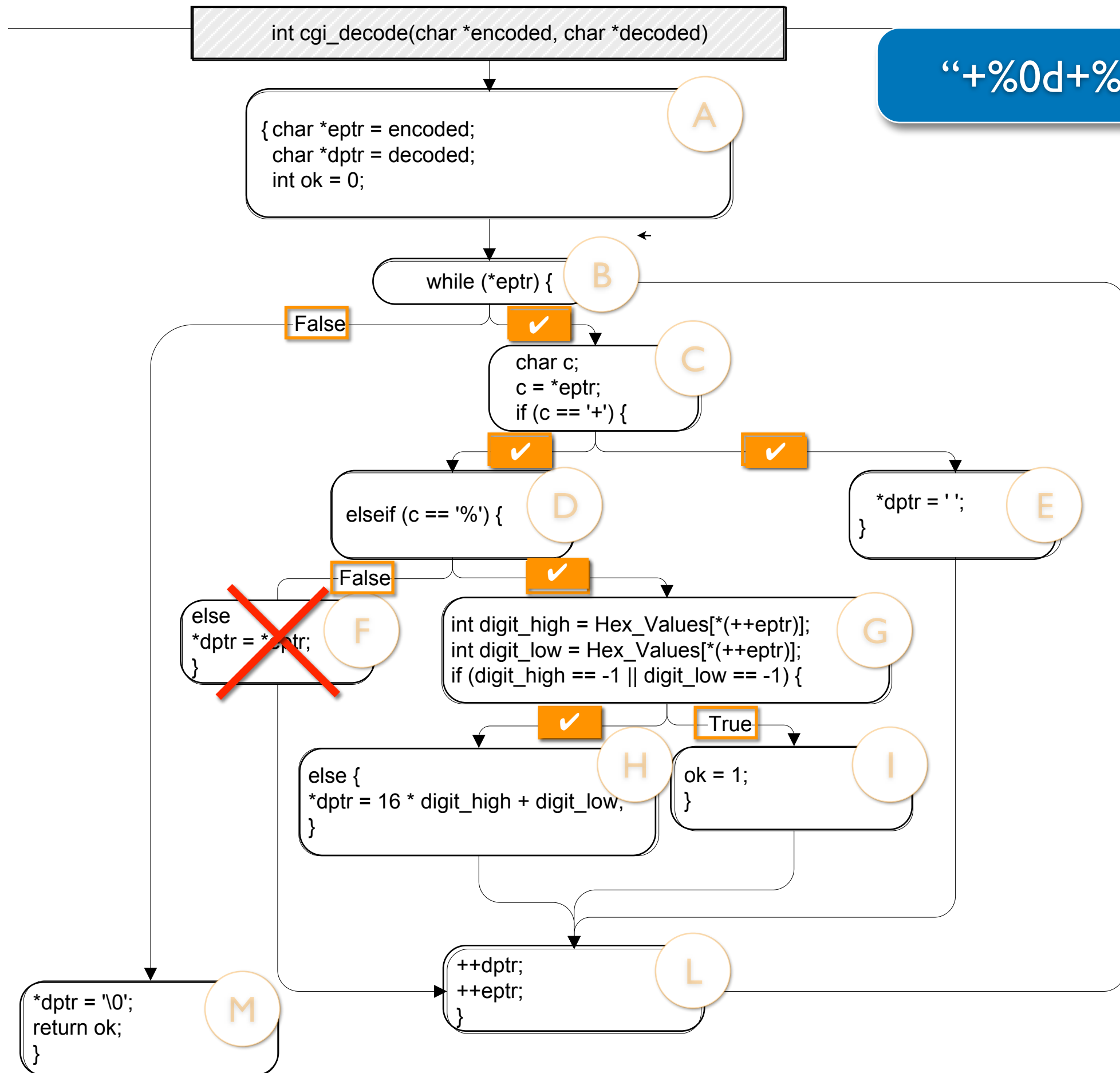


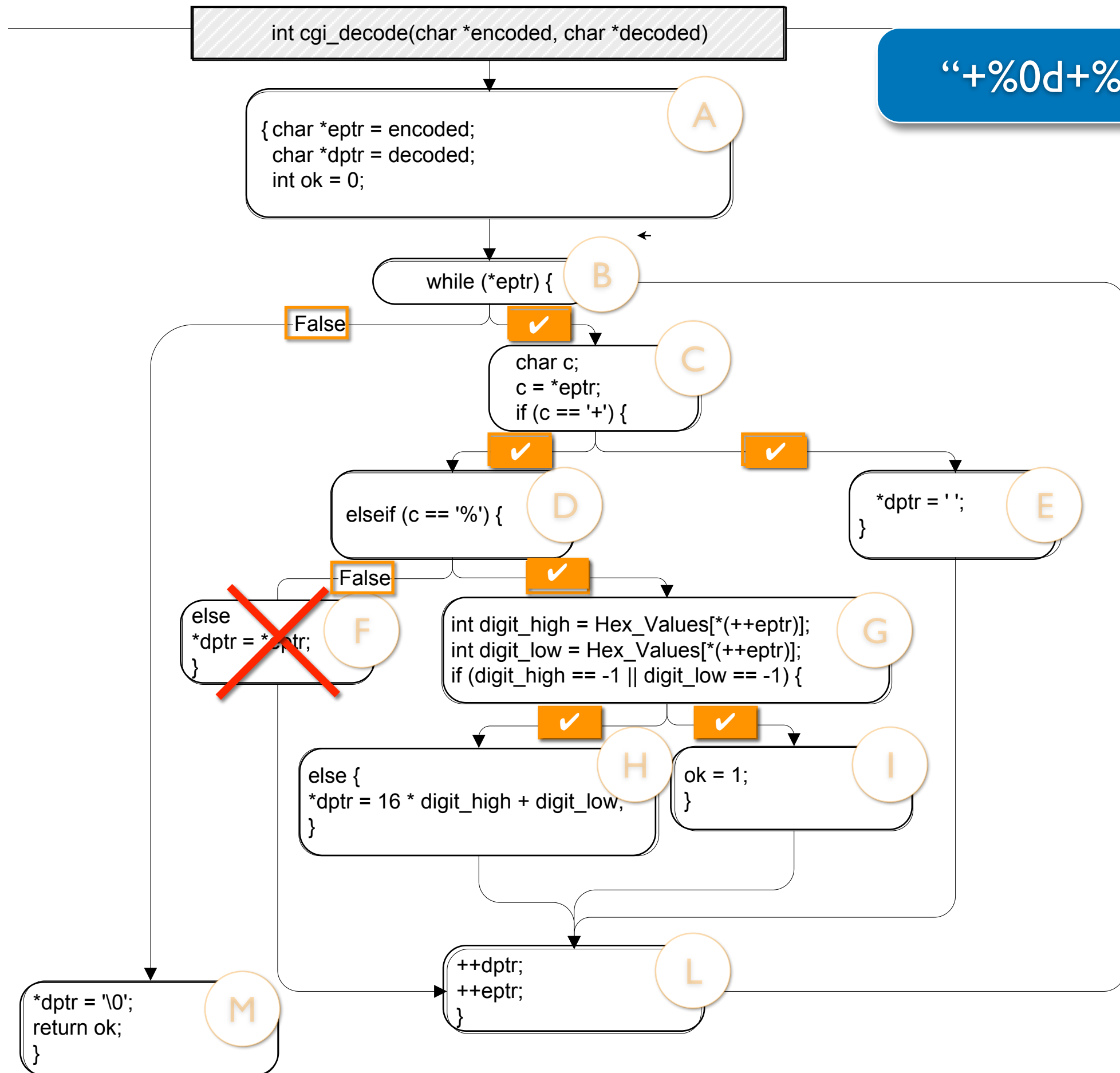


“+ %0d+ %4j”

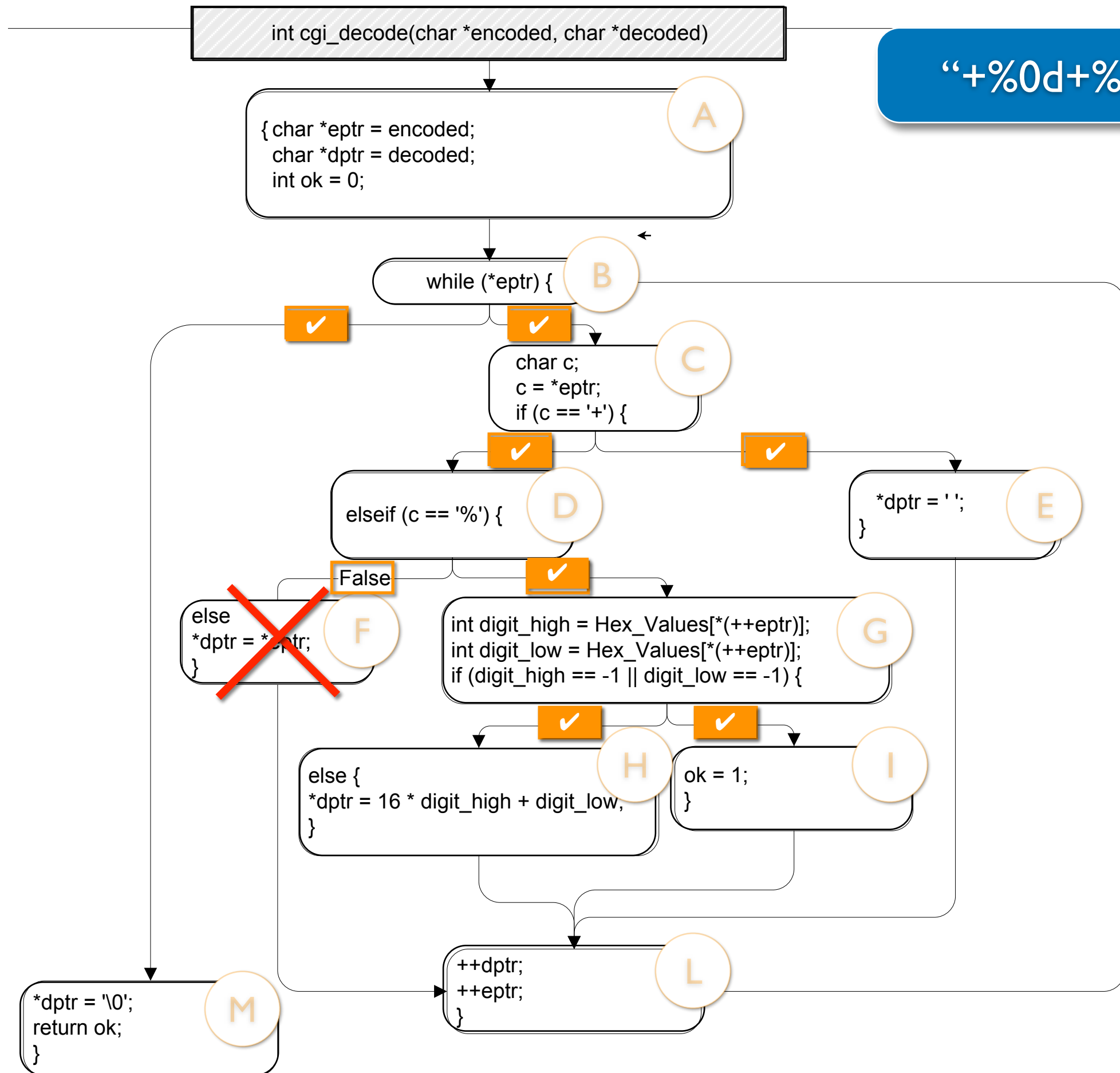


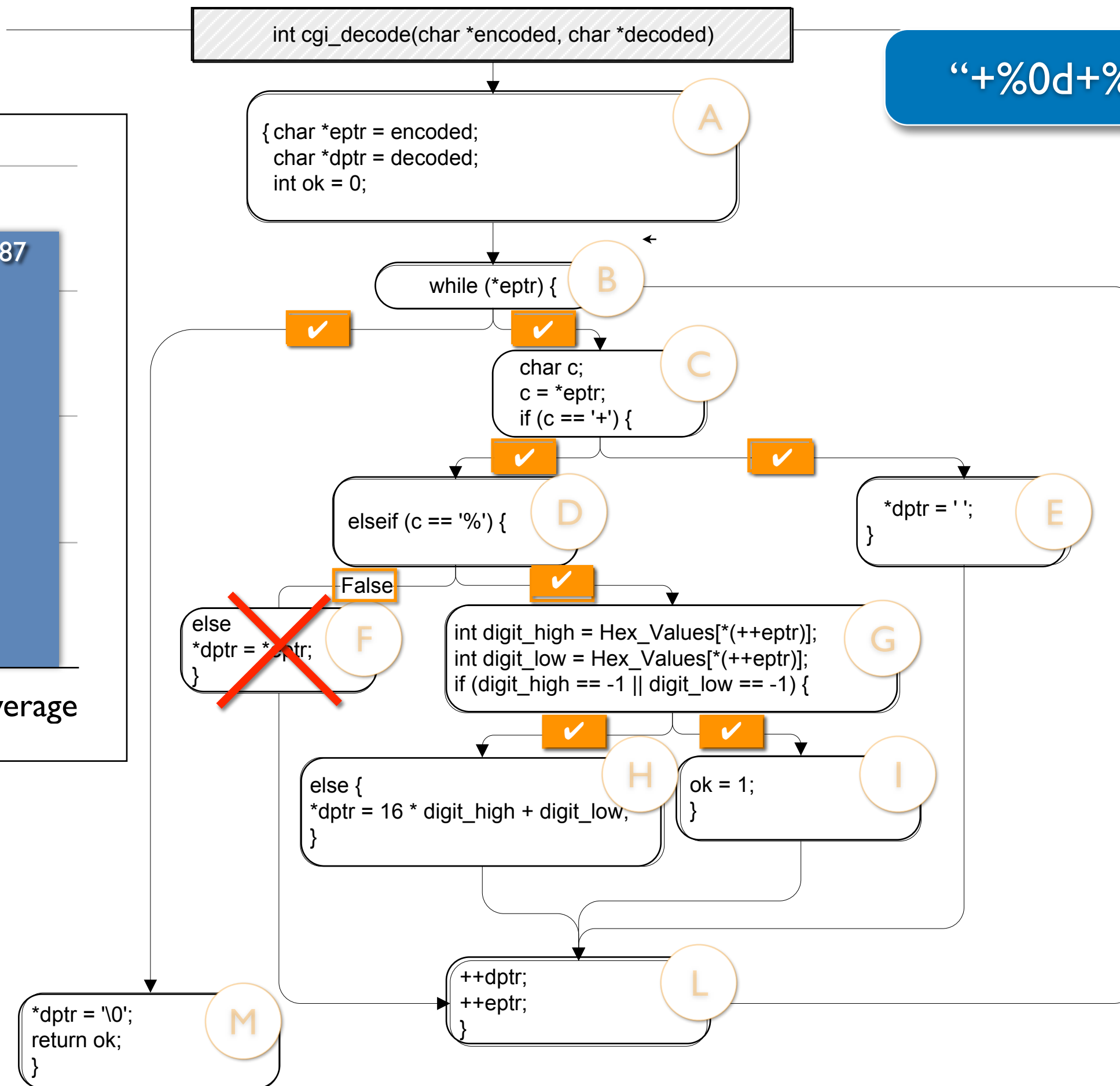
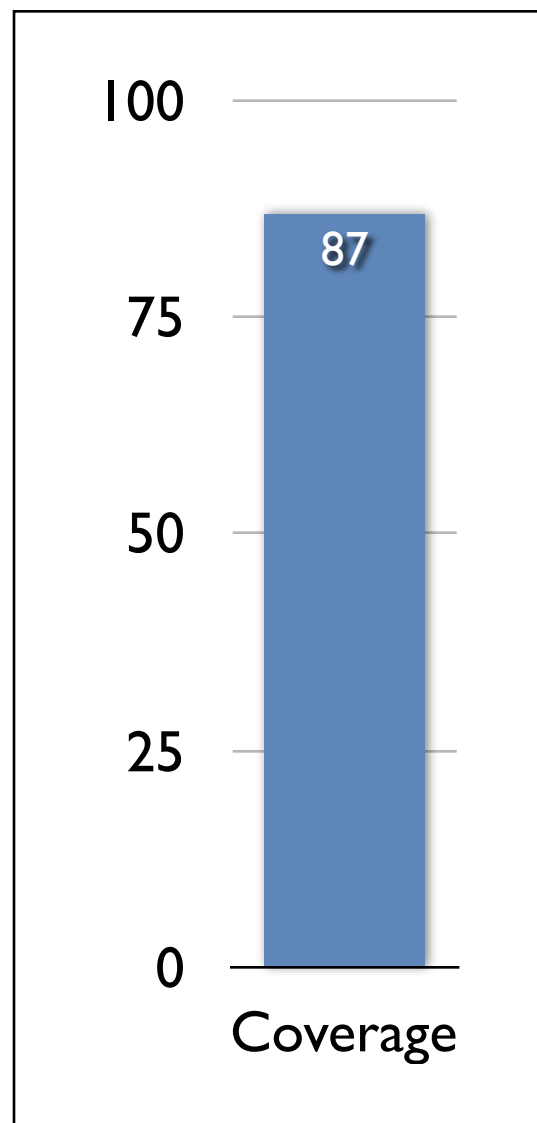




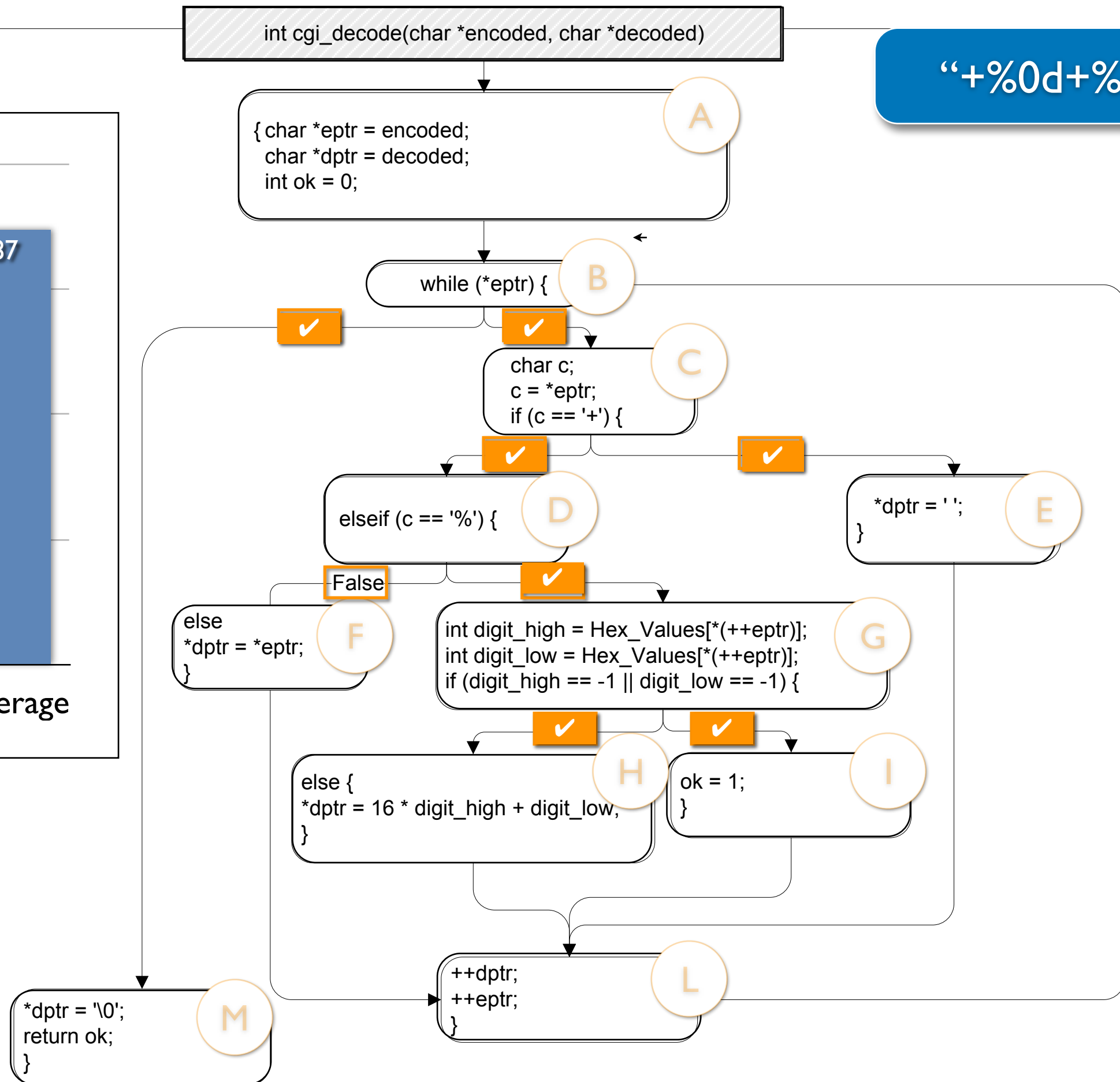
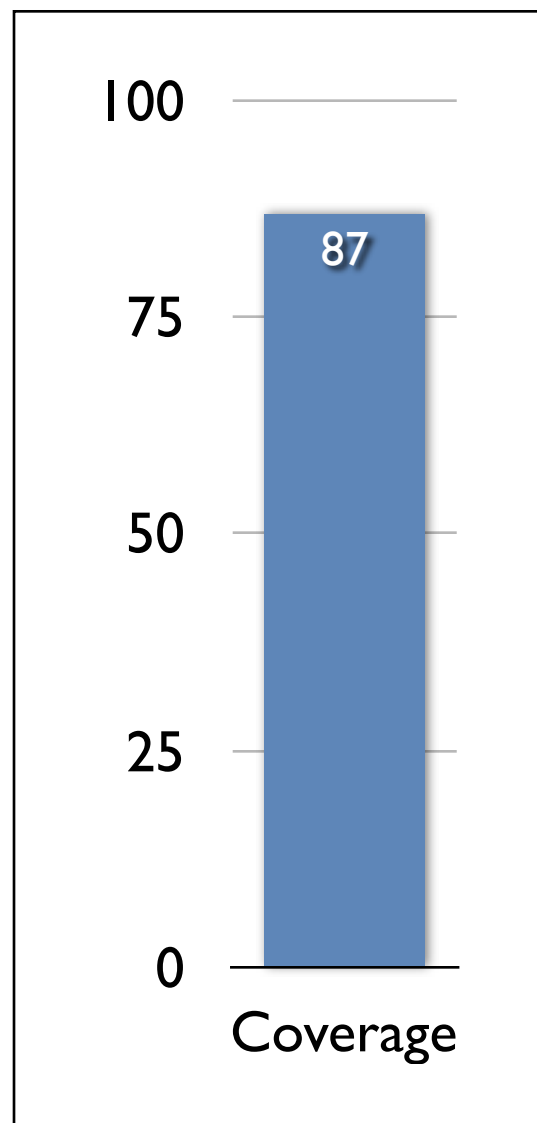


“+ %0d + %4j”



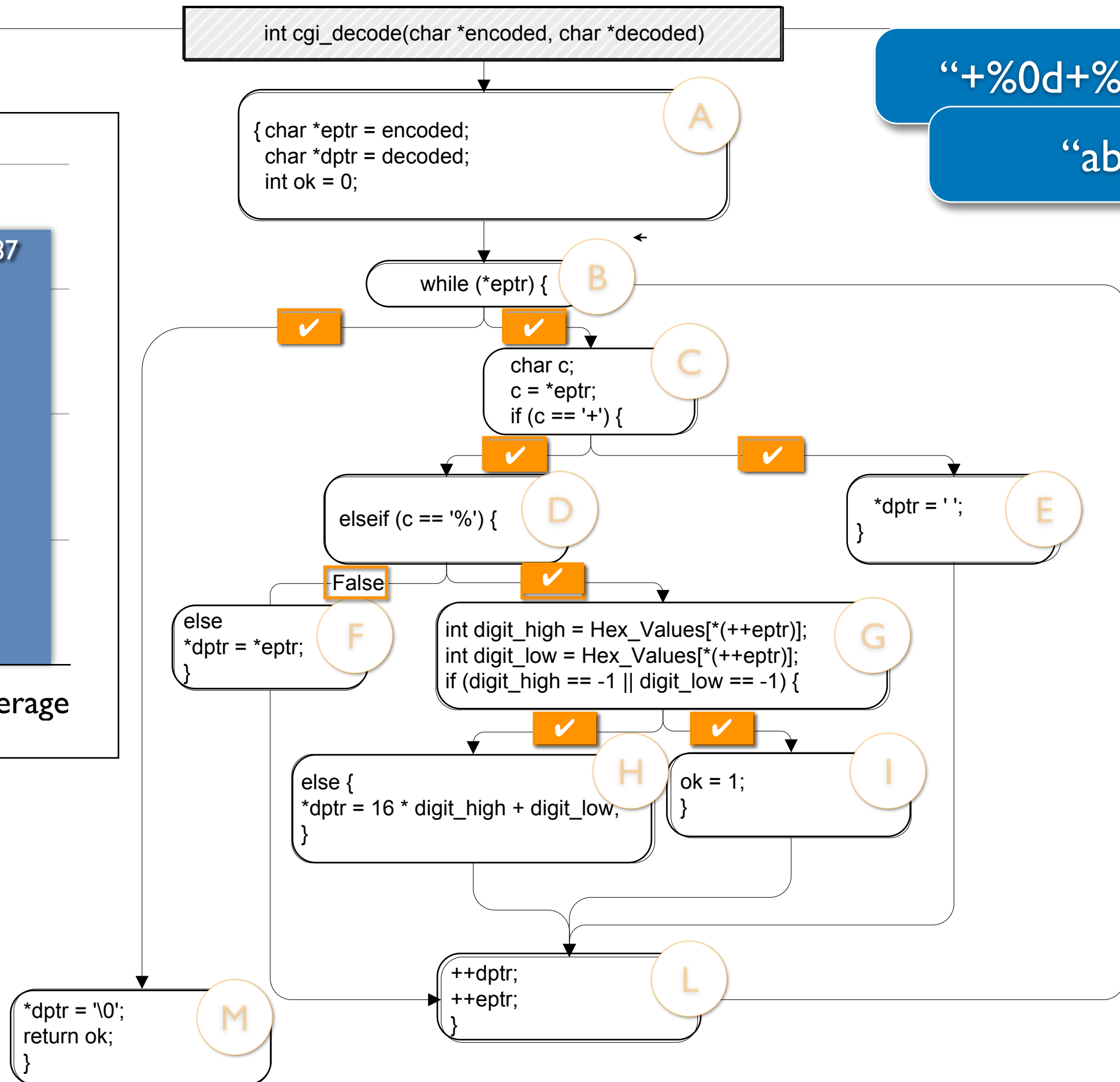
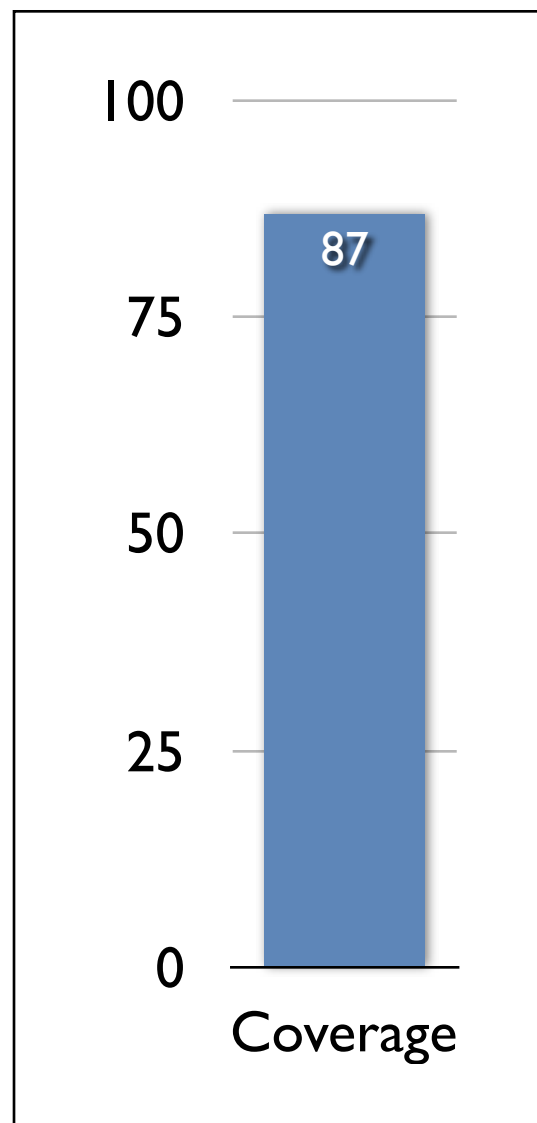


“+ %0d+ %4j”



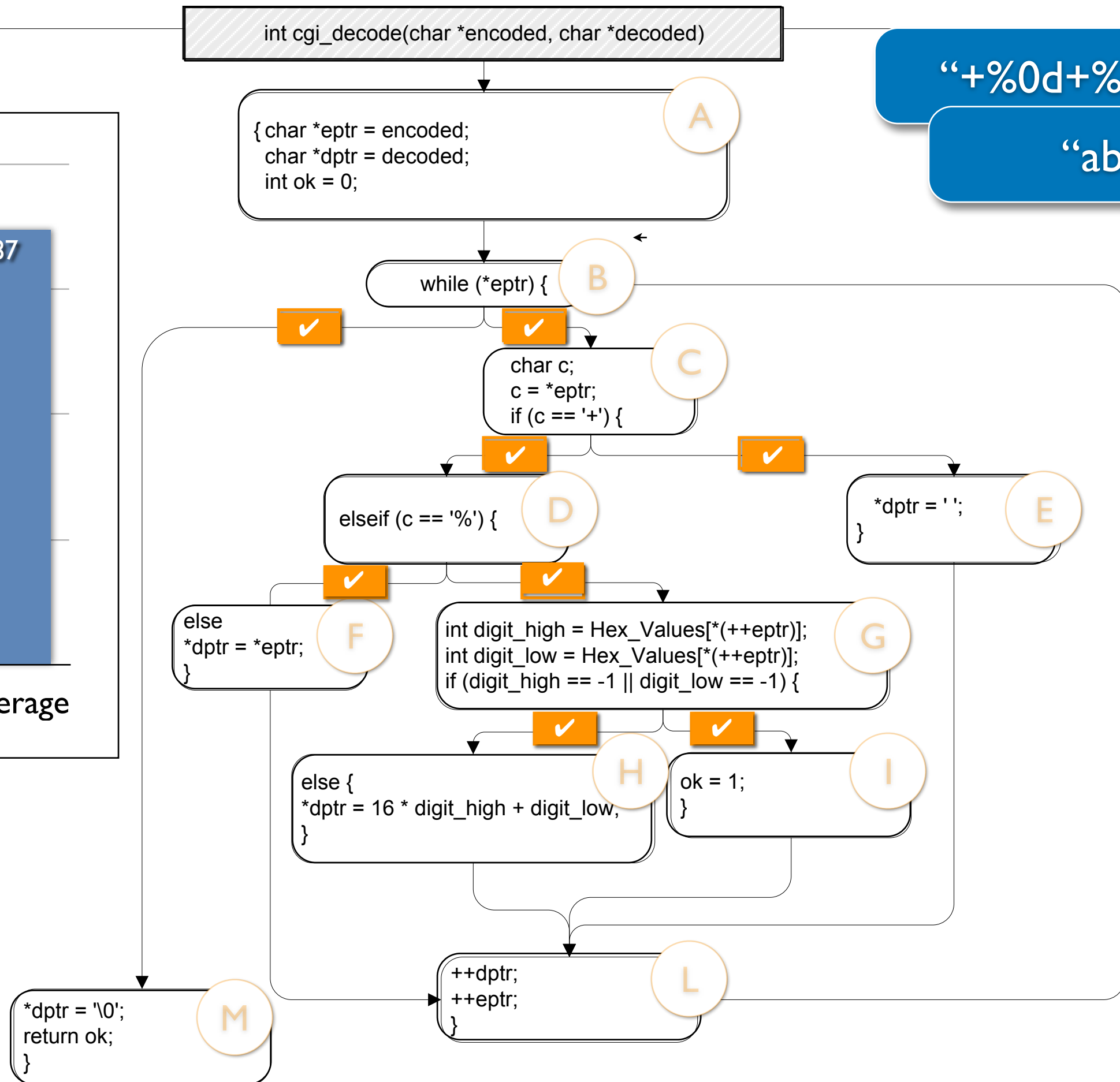
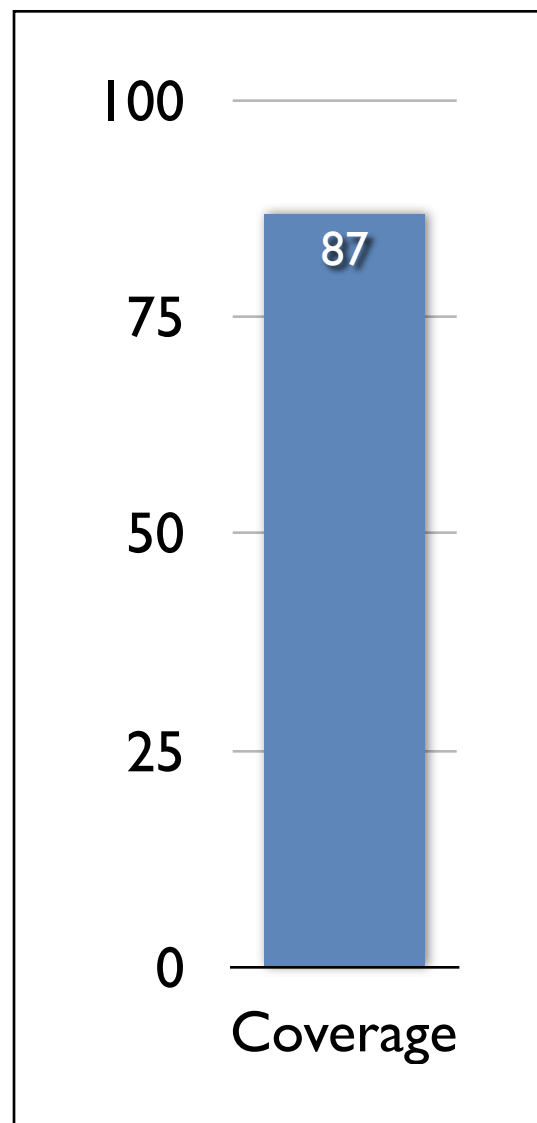
“+ %0d+ %4j”





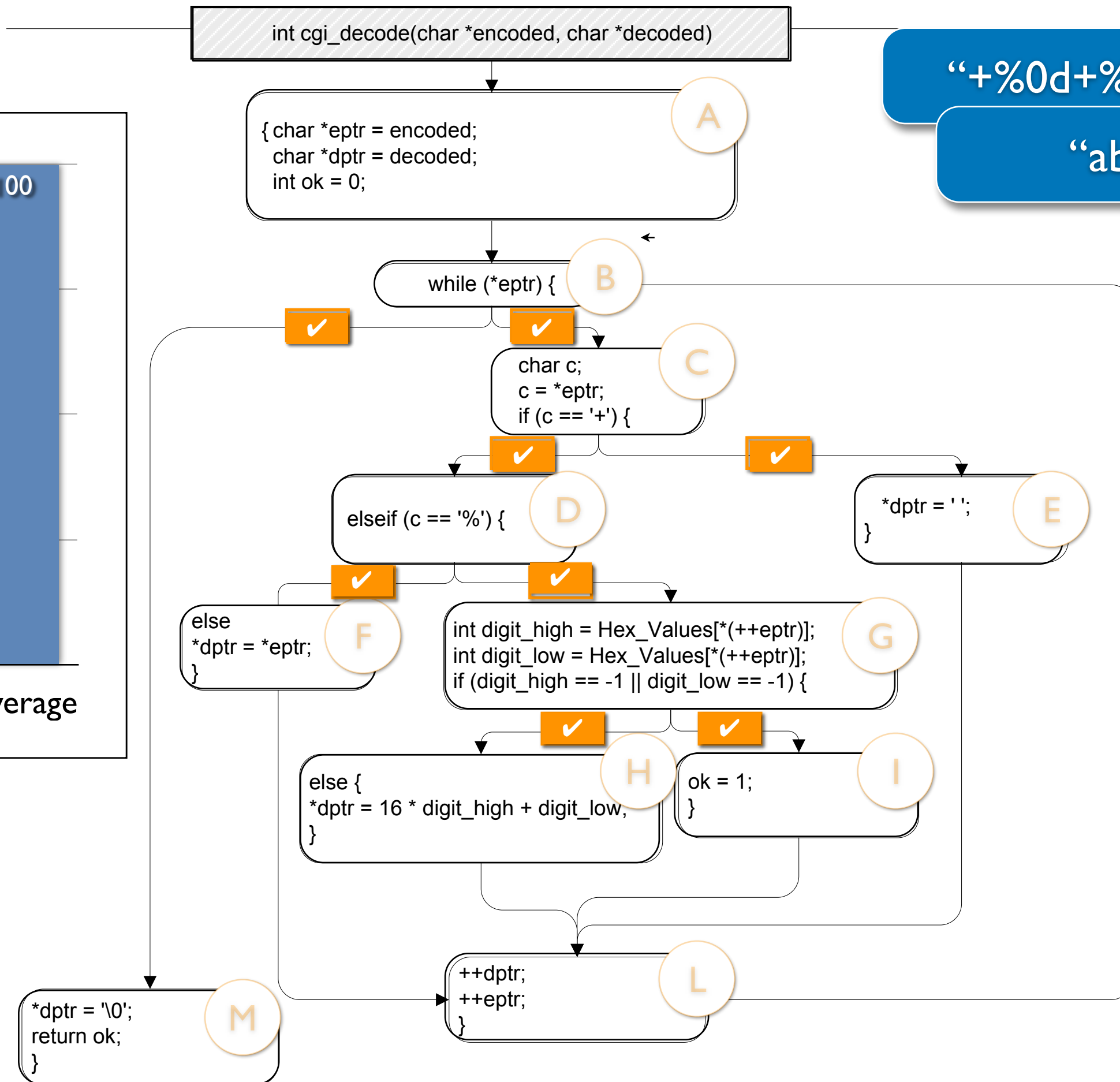
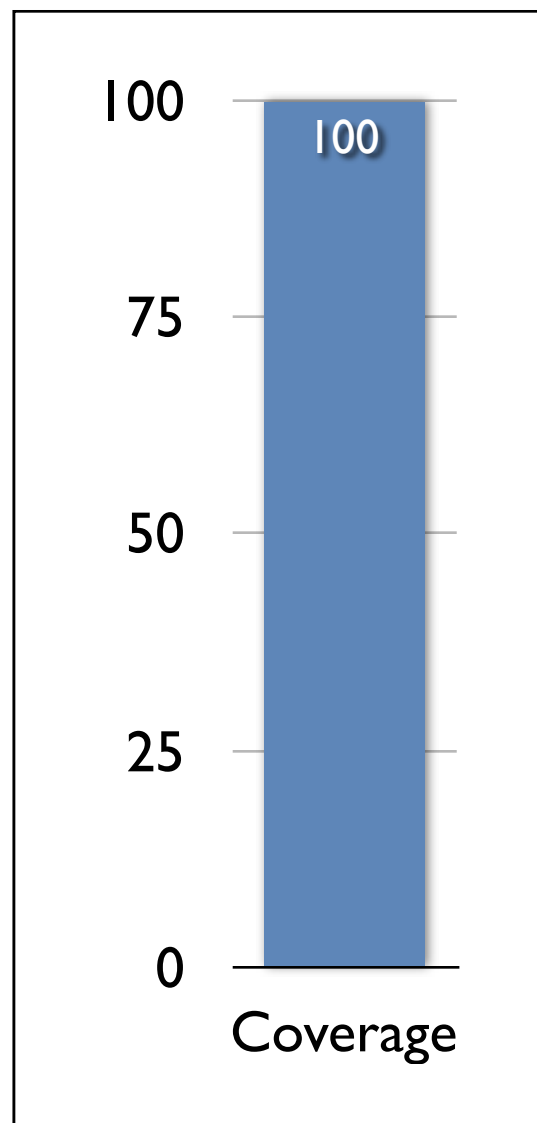
“+ %0d+ %4j”

“abc”



“+ %0d+ %4j”

“abc”



# Branch Coverage (Recubrimiento de Ramas de Programa?)

- Criterio de adecuación: Cada rama (*branch*) del programa debe ser ejecutada al menos una vez
- Recubrimiento: 
$$\frac{\# \text{ ramas ejecutadas}}{\# \text{ ramas}}$$
- Subsume el criterio de instrucciones de código  
*Visitar todas las ramas implica visitar todos los nodos*
- El criterio más aplicado en práctica
- Criterio estructural
  - Alternativa: *mutation scores*

# Search-based Testing (SBST)

## Combinatorial Interaction Testing

M.B. Cohen, M.B. Dwyer and J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints”, ISSTA 2007.

## Stress Testing

L. C. Briand, Y. Labiche, and M. Shousha, “Stress testing real-time systems with genetic algorithms”. GECCO 2005.

## State Machine Testing

K. Derderian, R. Hierons, M. Harman, and Q. Guo, “Automated unique input output sequence generation for conformance testing of FSMs”. The Computer Journal, 2006.



# Search-based Testing (SBST)



**Facebook Academics**

January 10, 2017 · 🌐

We're excited to announce that the team behind MaJiCke will be joining us at Facebook in London. MaJiCke has developed software that uses Search Based Software Engineering (SBSE) to help engineers find bugs while reducing the inefficiencies of writing test code. Their key product, Sapienz, is a multi-objective end-to-end testing system that automatically generates test sequences using SBSE to find crashes using the shortest path it can find.

The company's three co-founders Mark Harman (Scientific Advisor), Yue Jia (CEO), and Ke Mao (CTO) are researchers at University College London (UCL), currently funded, in part, by the UK's Engineering and Physical Sciences Research Council (EPSRC). They are all leaders in the field of computational search intelligence and will be joining an existing roster of strong engineering talent in our London office that is critical to building Facebook. We can't wait for the team to get started and to help us move faster towards our goal of connecting the world.



UNIVERSITY OF  
LEICESTER

# Generación Basada en Búsqueda

## Búsqueda Aleatoria

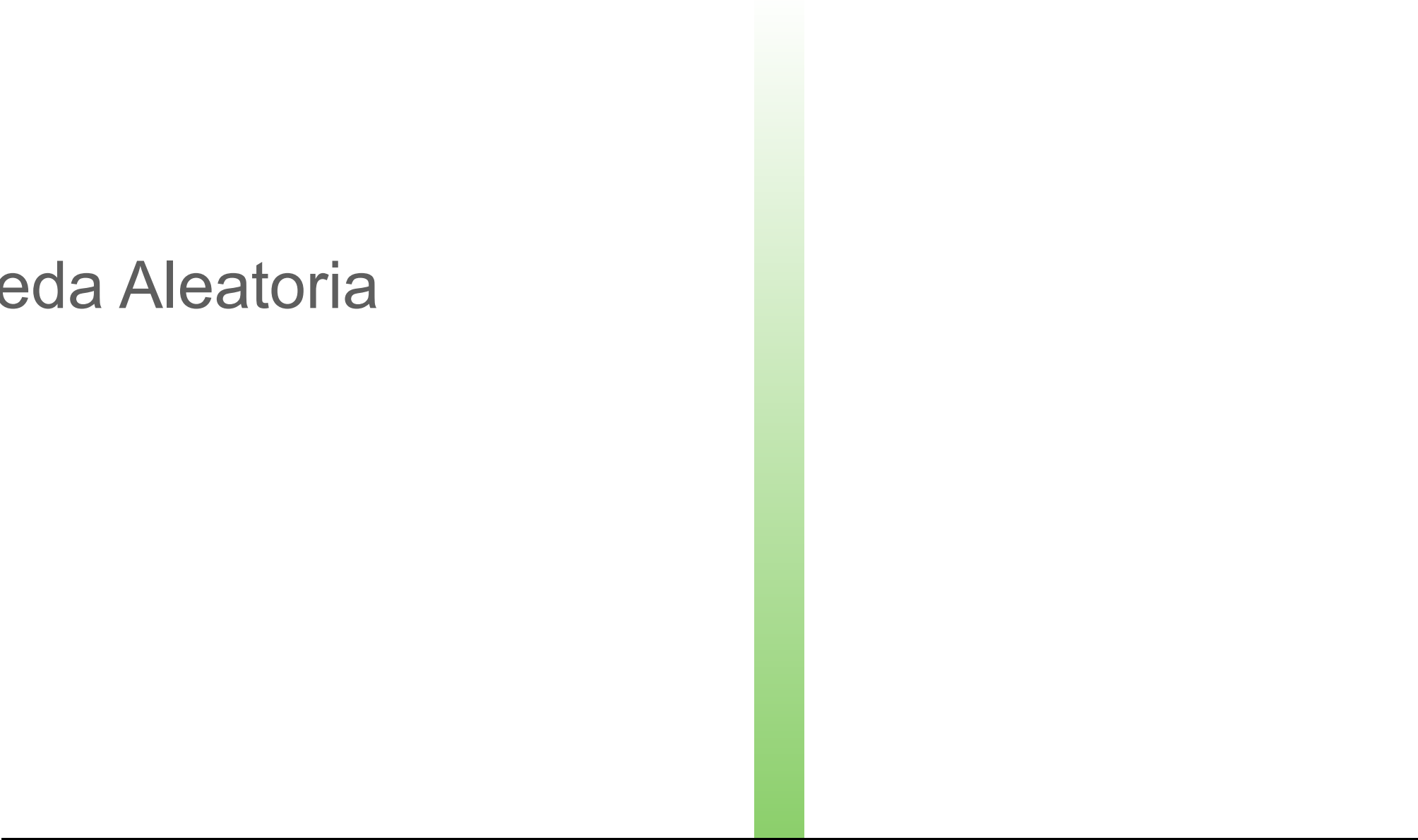
---

Entrada

C. Pacheco, S. K. Lahiri, M. D. Ernst and T. Ball, “Feedback-Directed Random Test Generation”, ICSE 2007.

# Generación Basada en Búsqueda

Búsqueda Aleatoria

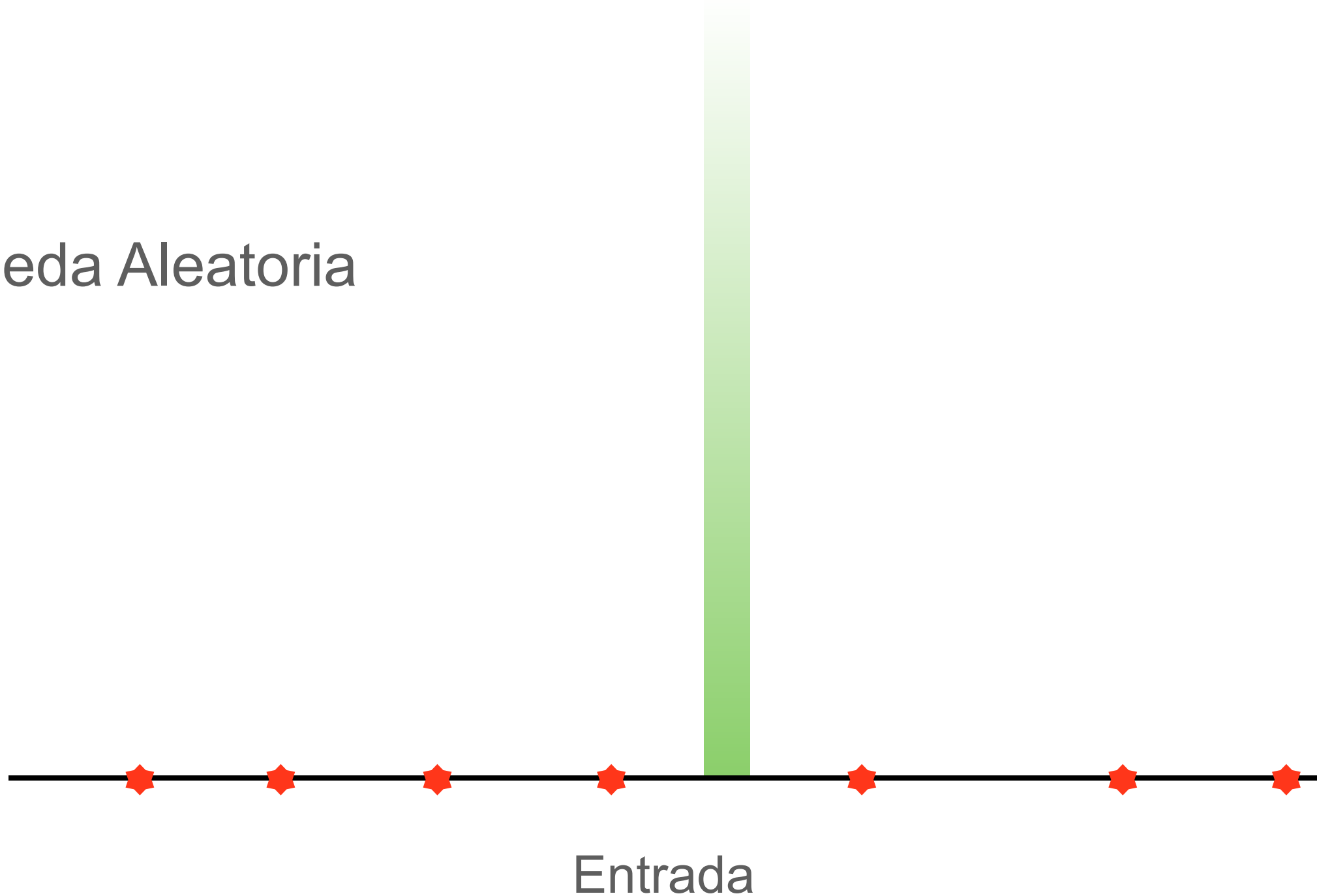


Entrada

C. Pacheco, S. K. Lahiri, M. D. Ernst and T. Ball, “Feedback-Directed Random Test Generation”, ICSE 2007.

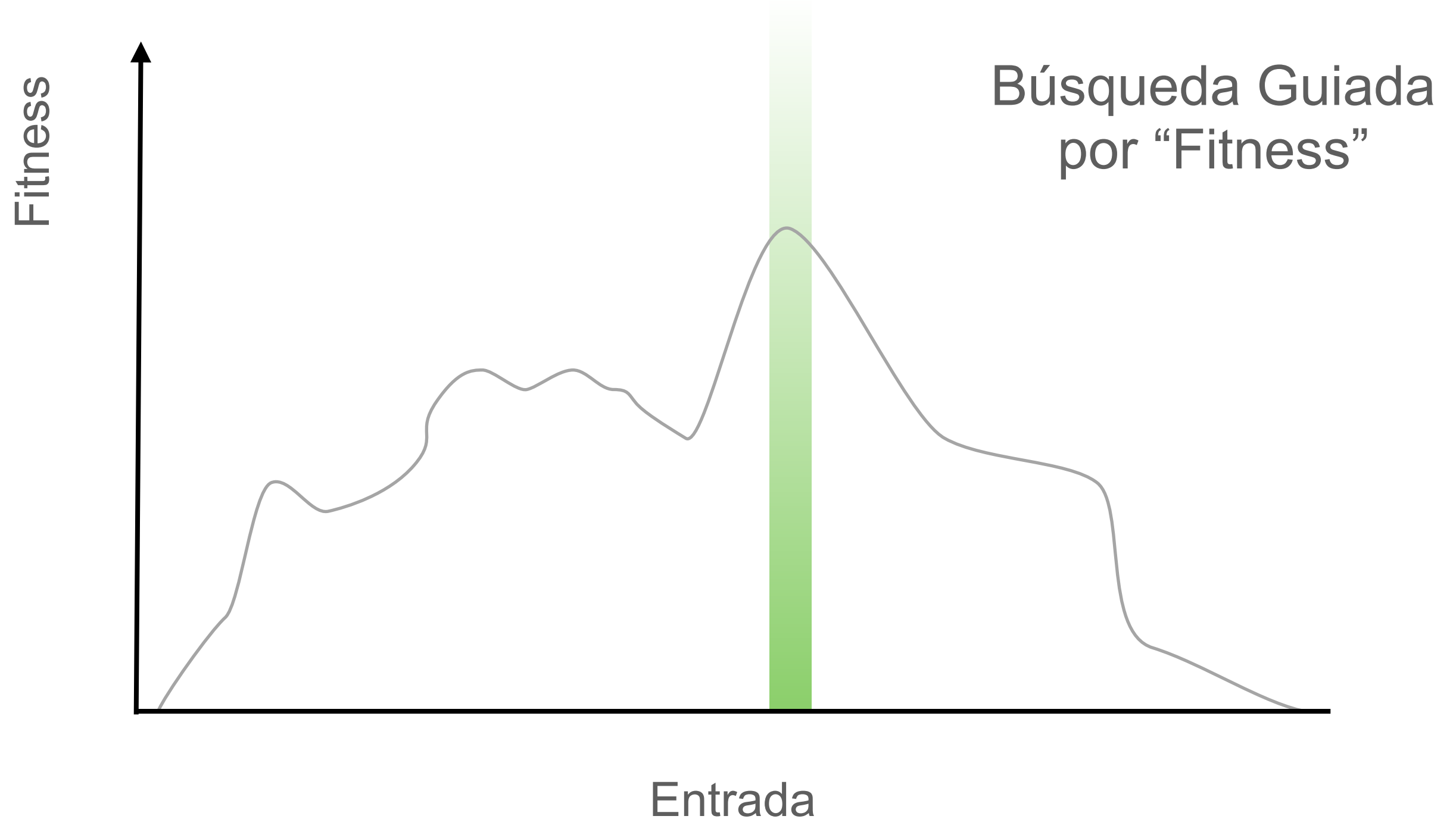
# Generación Basada en Búsqueda

Búsqueda Aleatoria

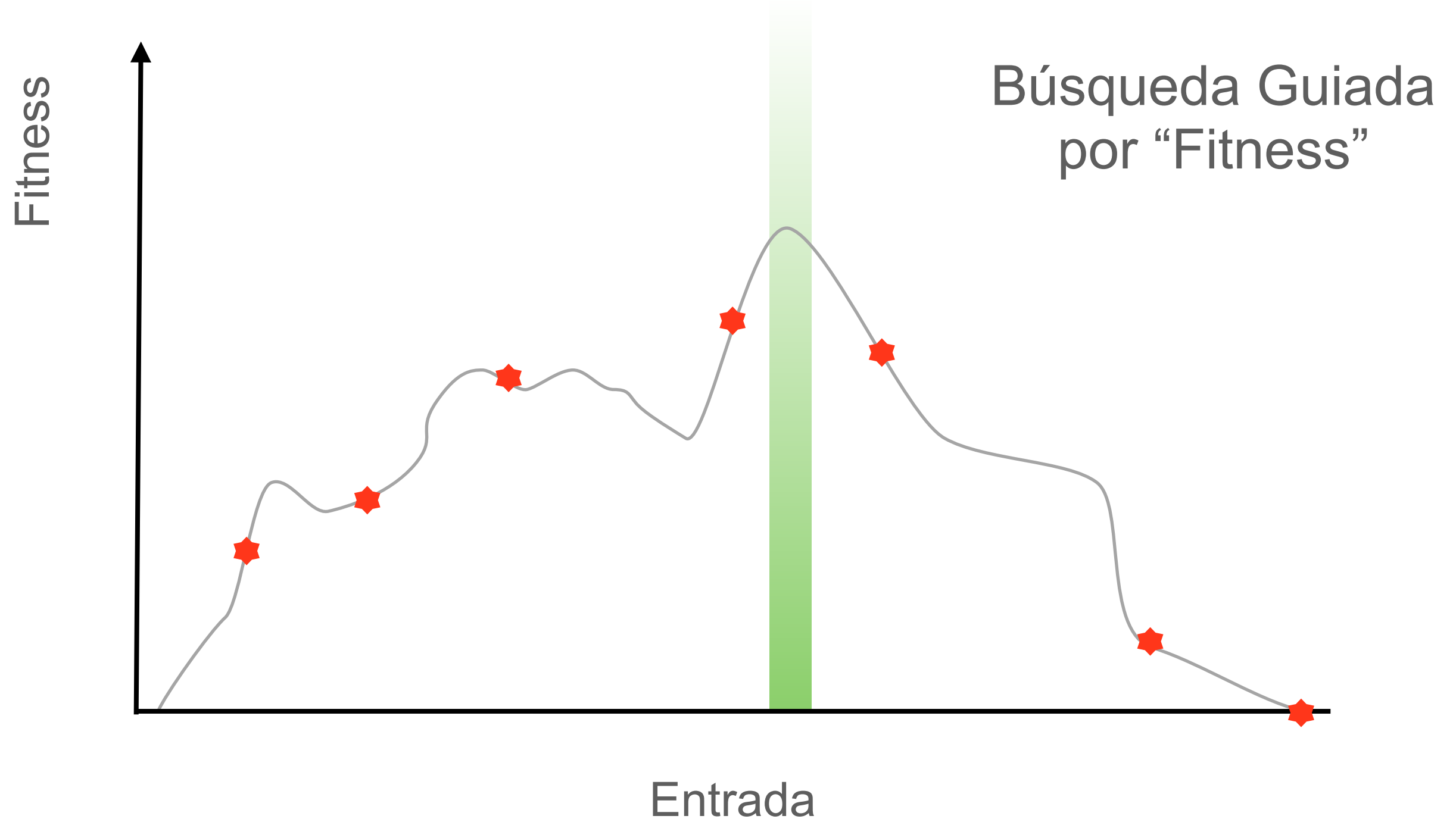


C. Pacheco, S. K. Lahiri, M. D. Ernst and T. Ball, “Feedback-Directed Random Test Generation”, ICSE 2007.

# Generación Basada en Búsqueda

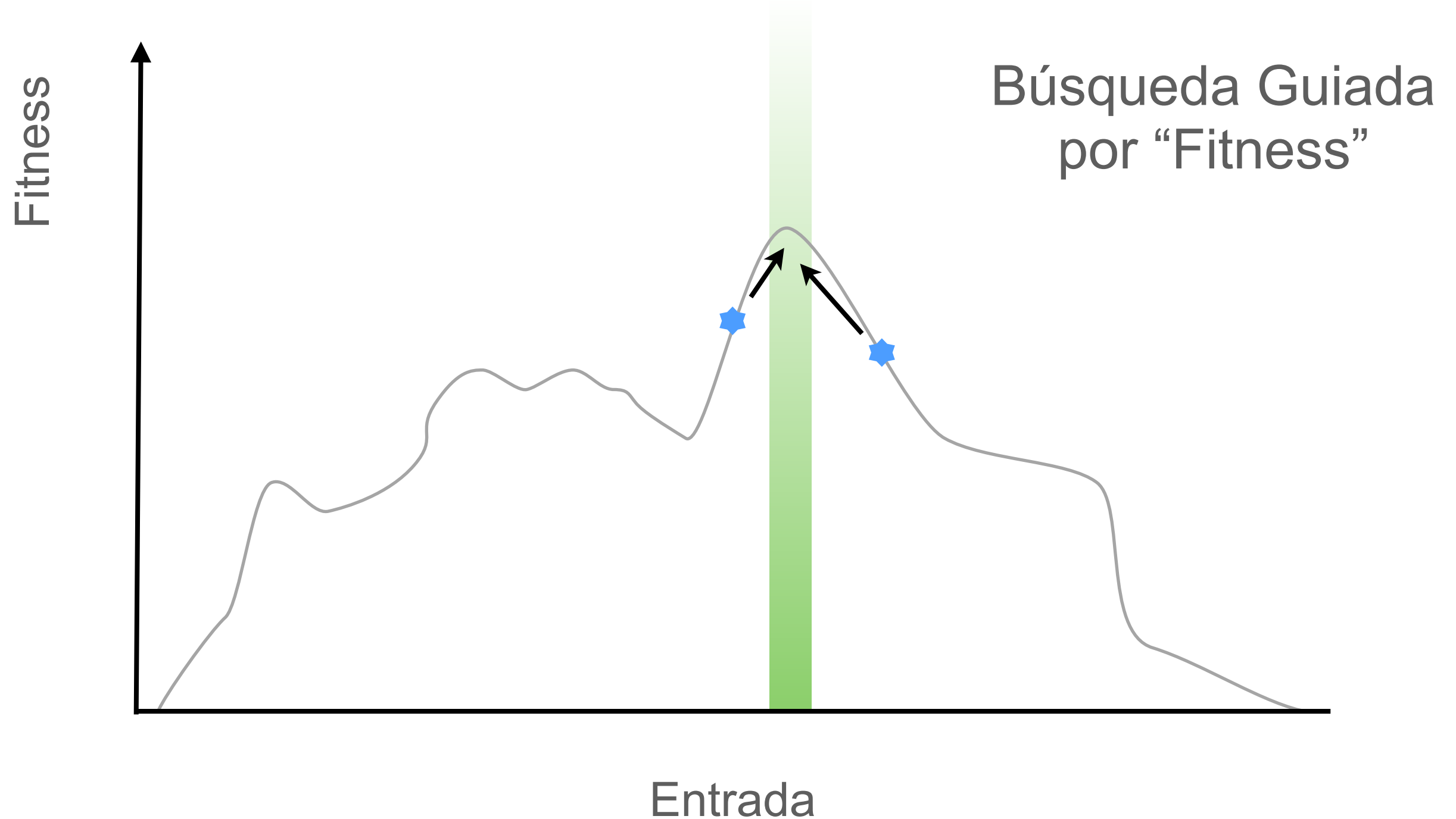


# Generación Basada en Búsqueda



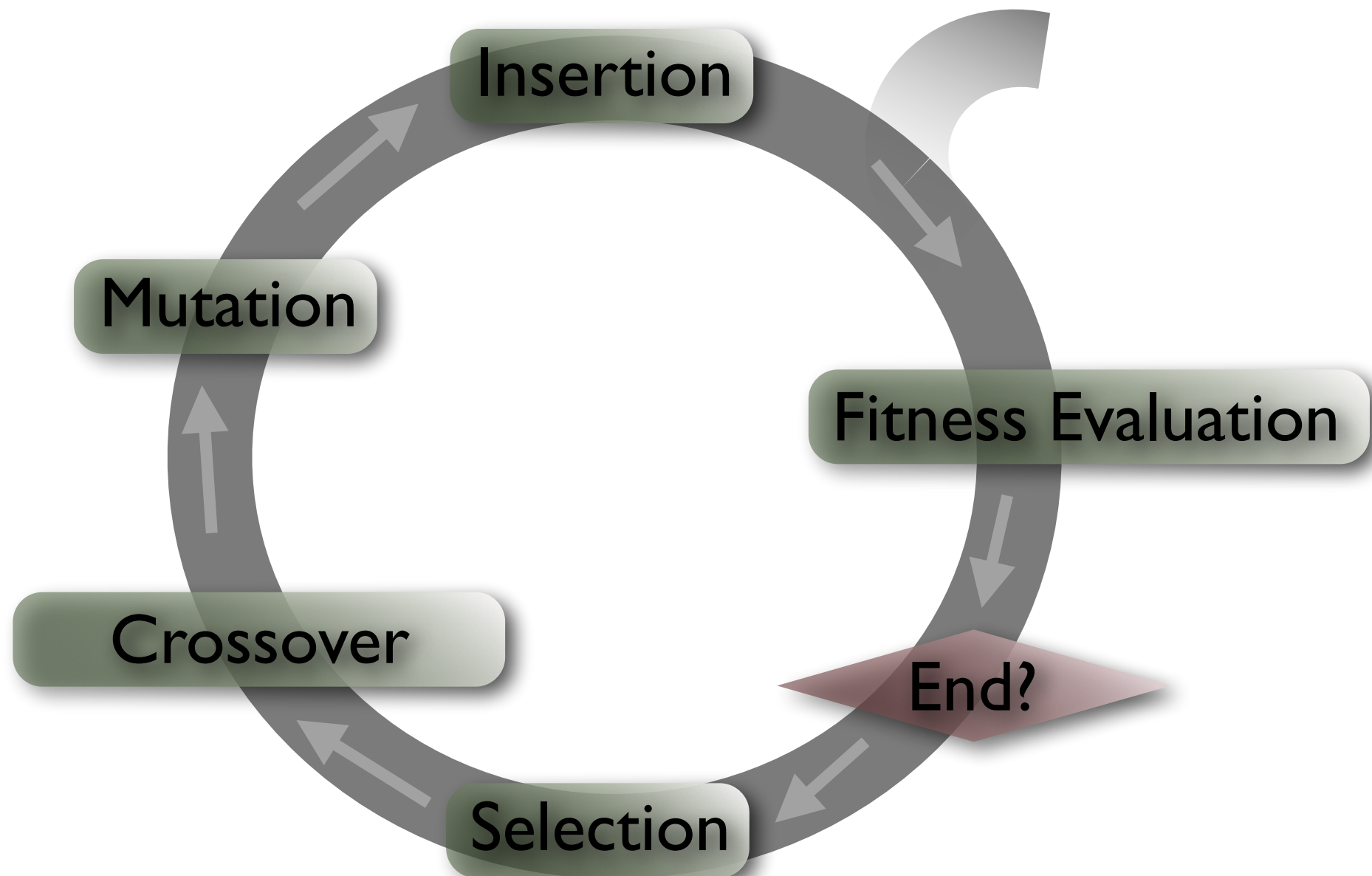


# Generación Basada en Búsqueda

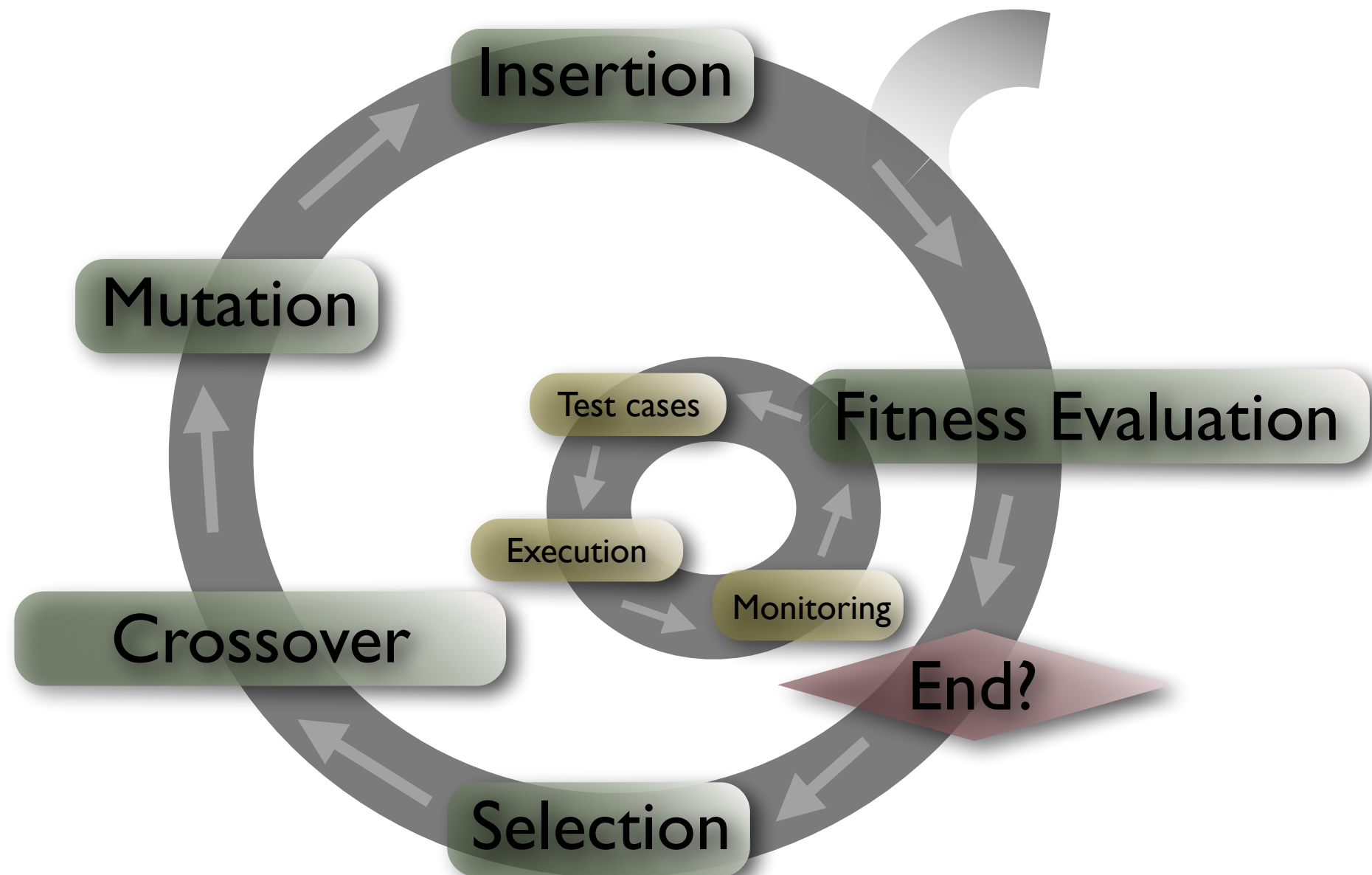


# Algoritmo Genético

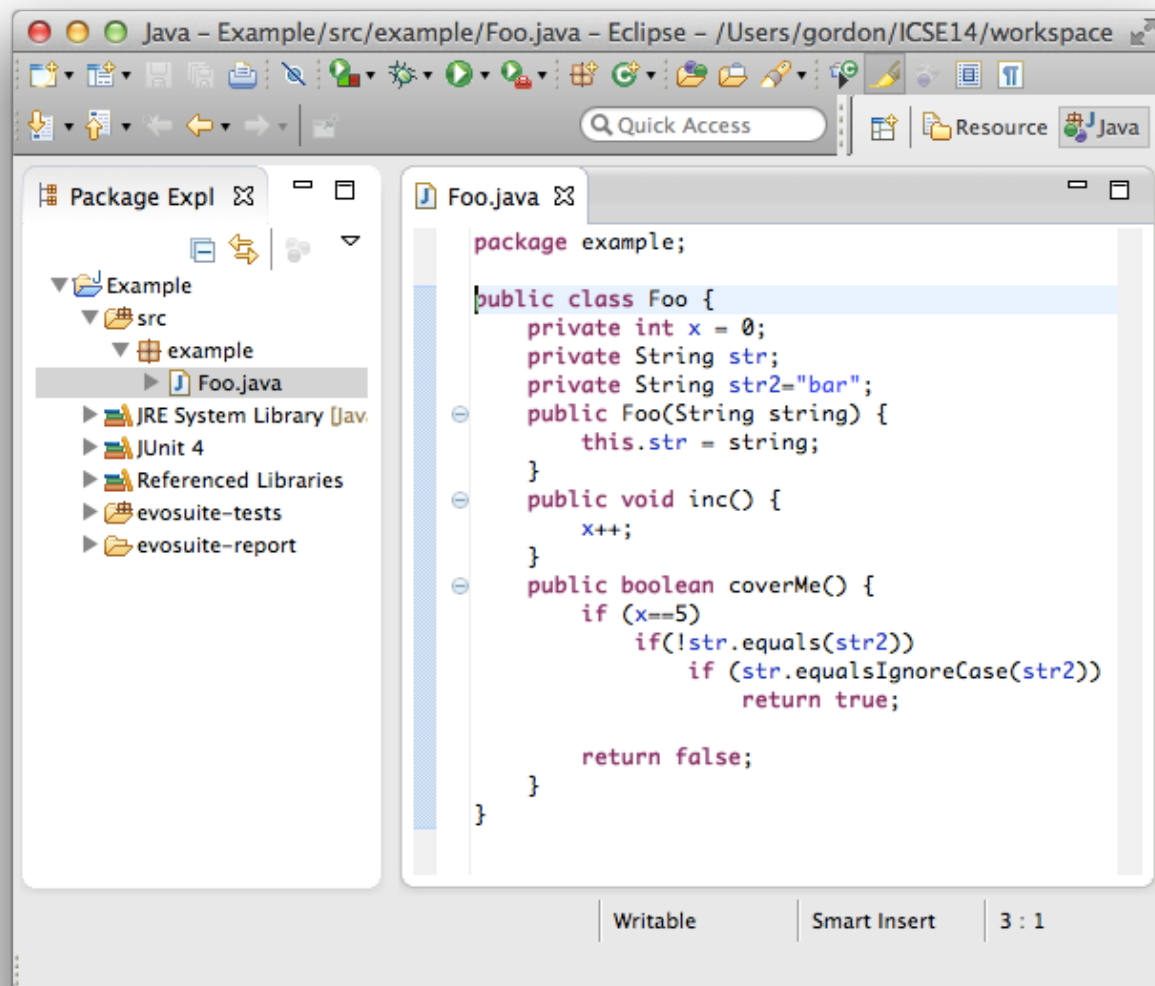
# Algoritmo Genético



# Algoritmo Genético

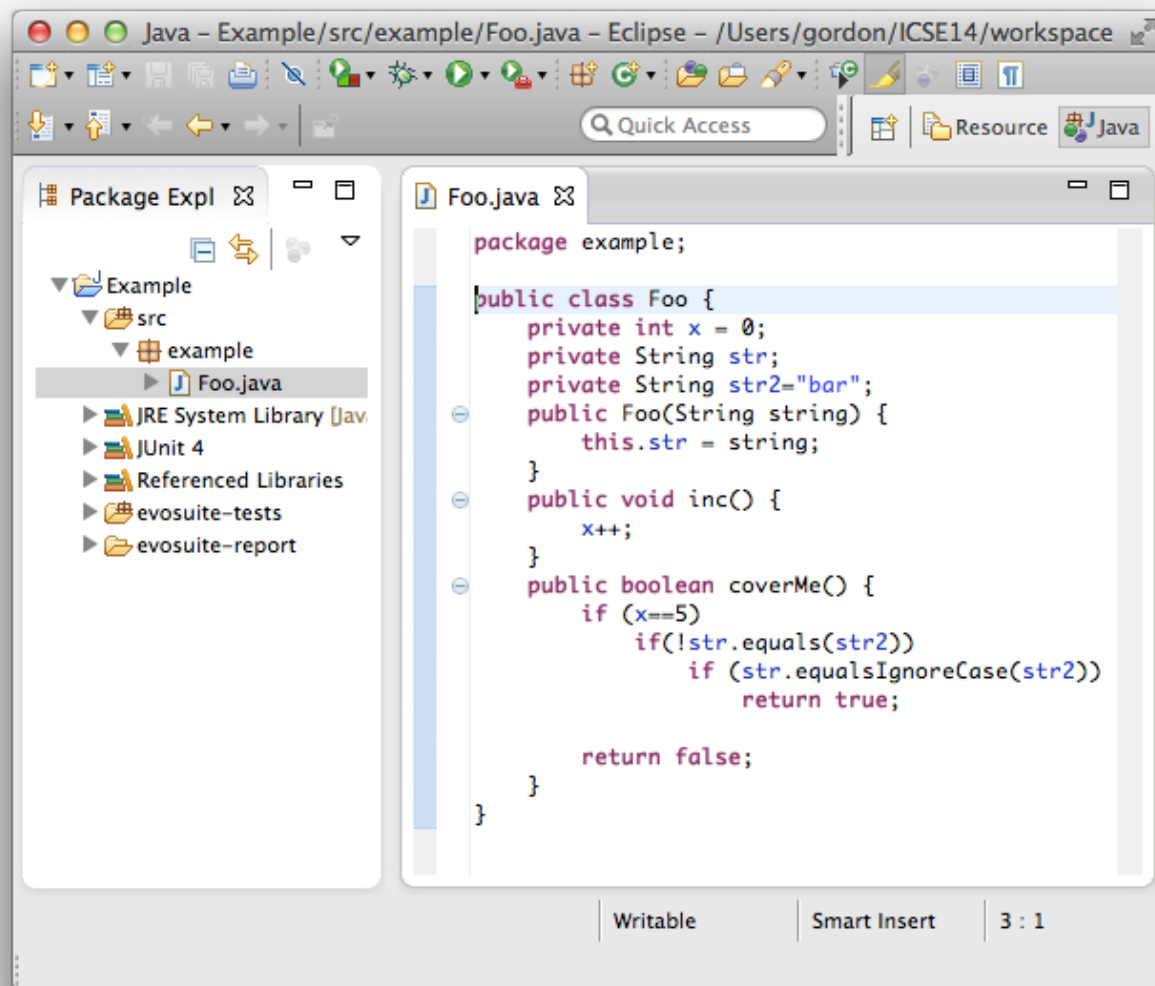


# Caso de Uso Básico



Código Fuente

# Caso de Uso Básico



The screenshot shows the Eclipse IDE with the Package Explorer on the left displaying the project structure: Example > src > example > Foo.java. The main editor displays the source code of Foo.java:

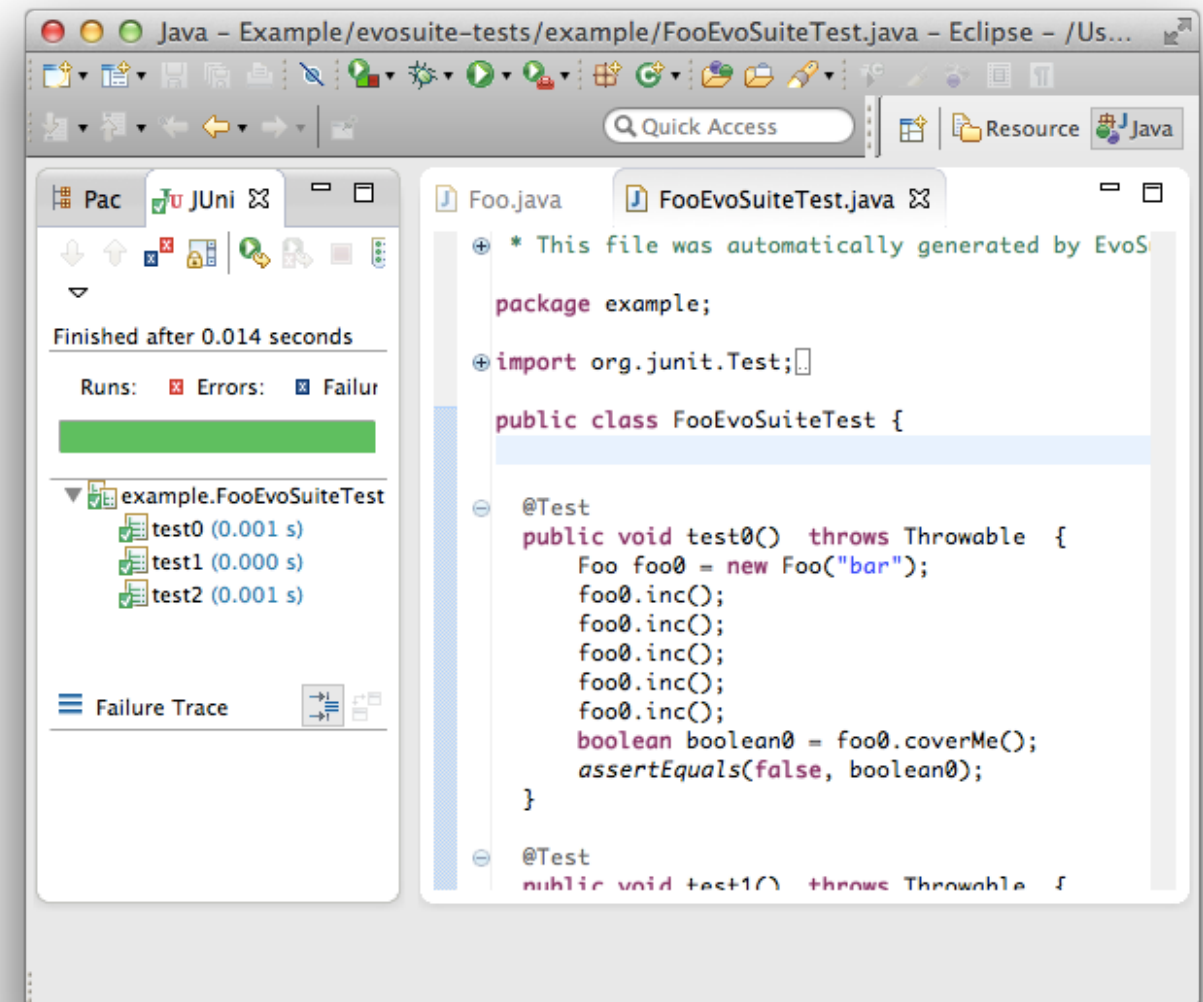
```
package example;

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;
    }
}
```

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '3 : 1'.

Código Fuente



The screenshot shows the Eclipse IDE with the Package Explorer on the left displaying the project structure: Example > evosuite-tests > example > FooEvoSuiteTest.java. The main editor displays the source code of FooEvoSuiteTest.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 {
```

The left sidebar shows the test results for example.FooEvoSuiteTest:

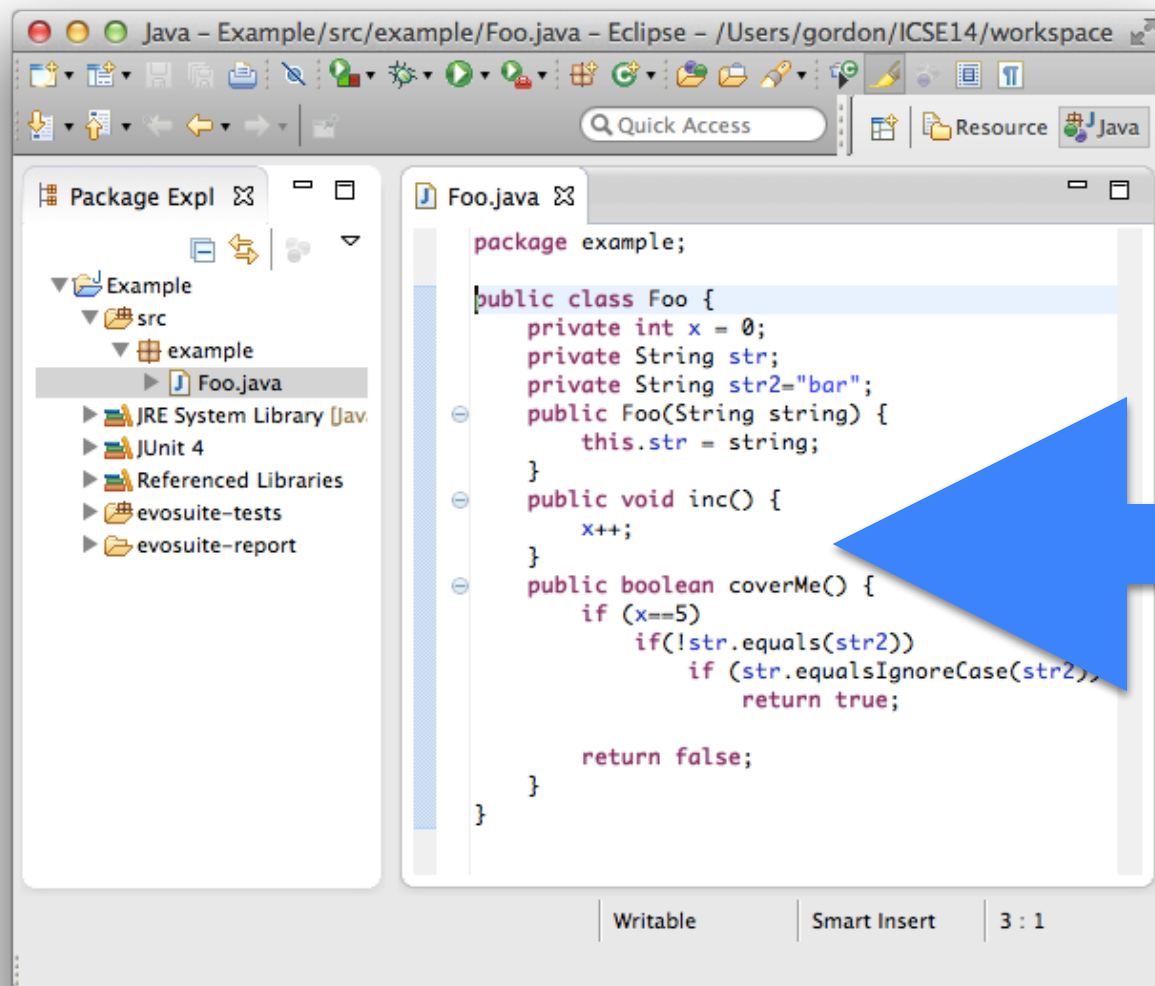
- test0 (0.001 s)
- test1 (0.000 s)
- test2 (0.001 s)

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '3 : 1'.

Casos de Prueba



# Caso de Uso Básico

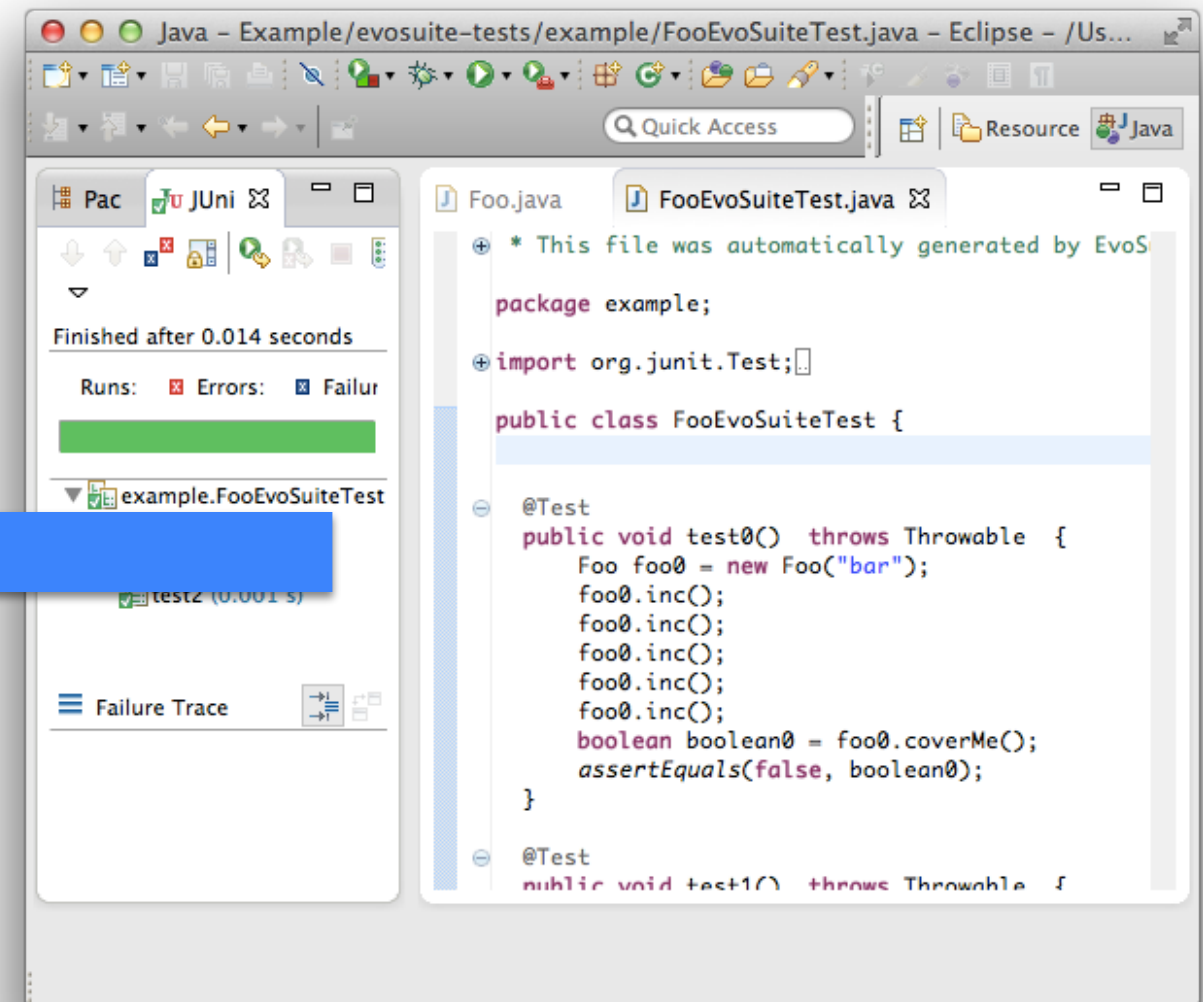


```
package example;

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;
    }
}
```

Código Fuente



```
* This file was automatically generated by EvoSuite

package example;

import org.junit.Test;

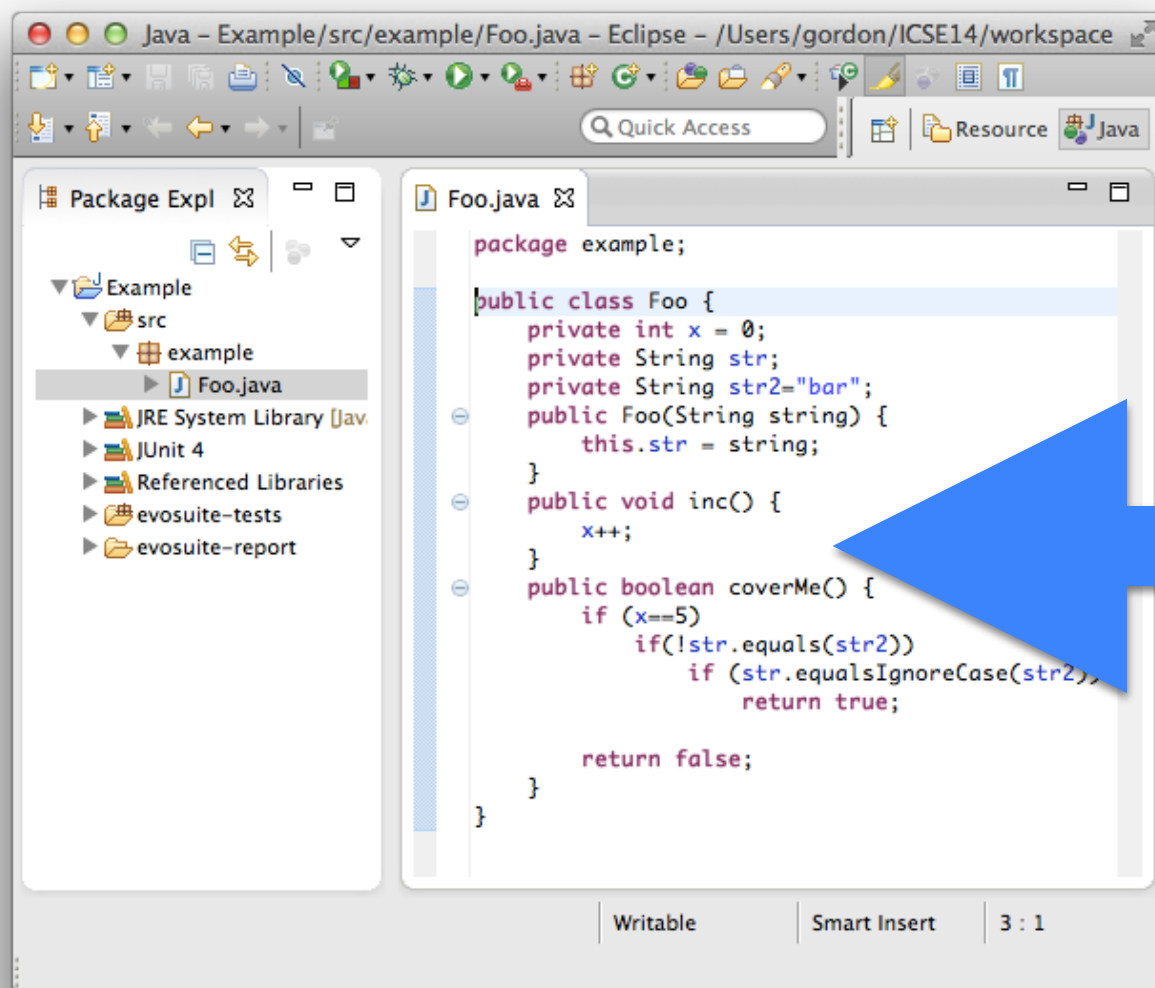
public class FooEvoSuiteTest {

    @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 {
```

Casos de Prueba

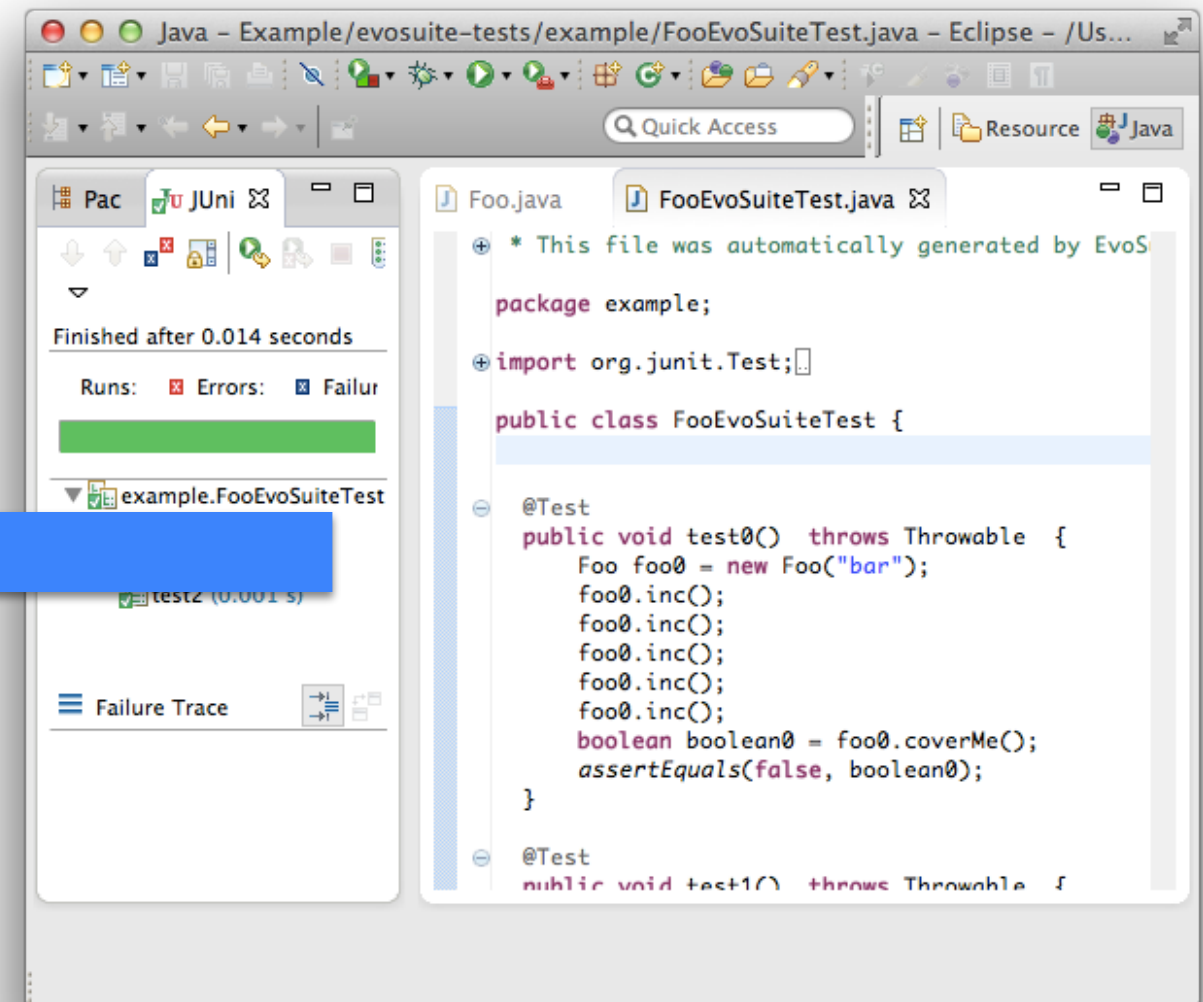
# Caso de Uso Básico



The screenshot shows the Eclipse IDE with the Package Explorer on the left displaying a project structure with 'Example' and 'src' folders. The main editor displays the source code of 'Foo.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;
    }
}
```



The screenshot shows the Eclipse IDE with the Package Explorer on the left displaying a project structure with 'Example' and 'src' folders. The main editor displays the source code of 'FooEvoSuiteTest.java', which is automatically generated by EvoSuite. A 'JUnit' tab is also visible, showing a successful test run.

```
* This file was automatically generated by EvoSuite

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 {
```

Código Fuente

Casos de Prueba

Generación Automática

@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();

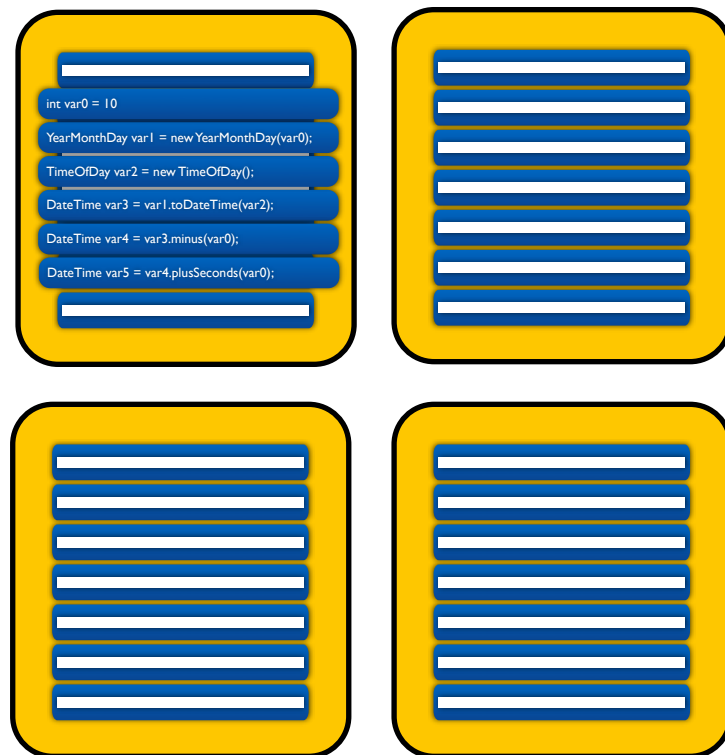
DateTime var3 = var1.toDateTime(var2);

DateTime var4 = var3.minus(var0);

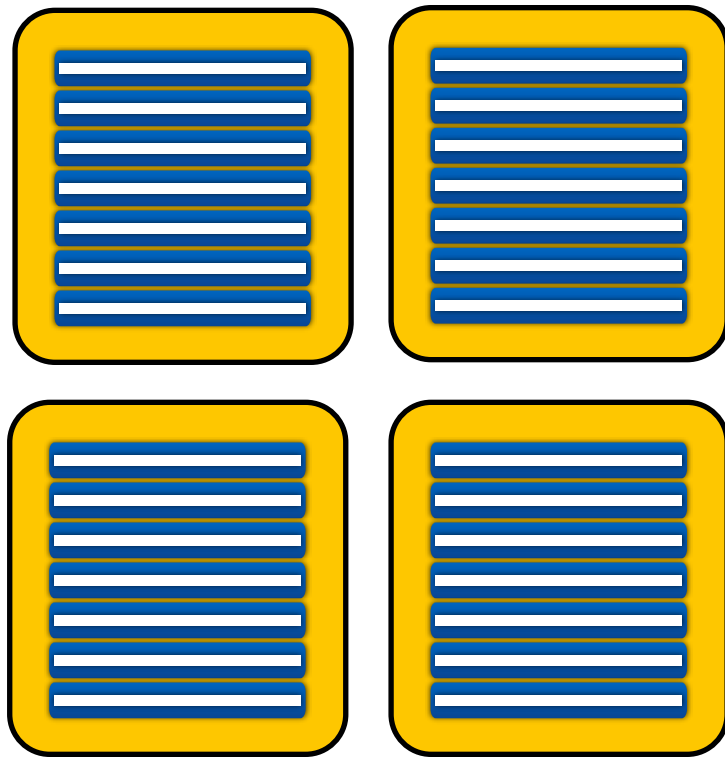
DateTime var5 = var4.plusSeconds(var0);

}

# Generación de “Test Suites”

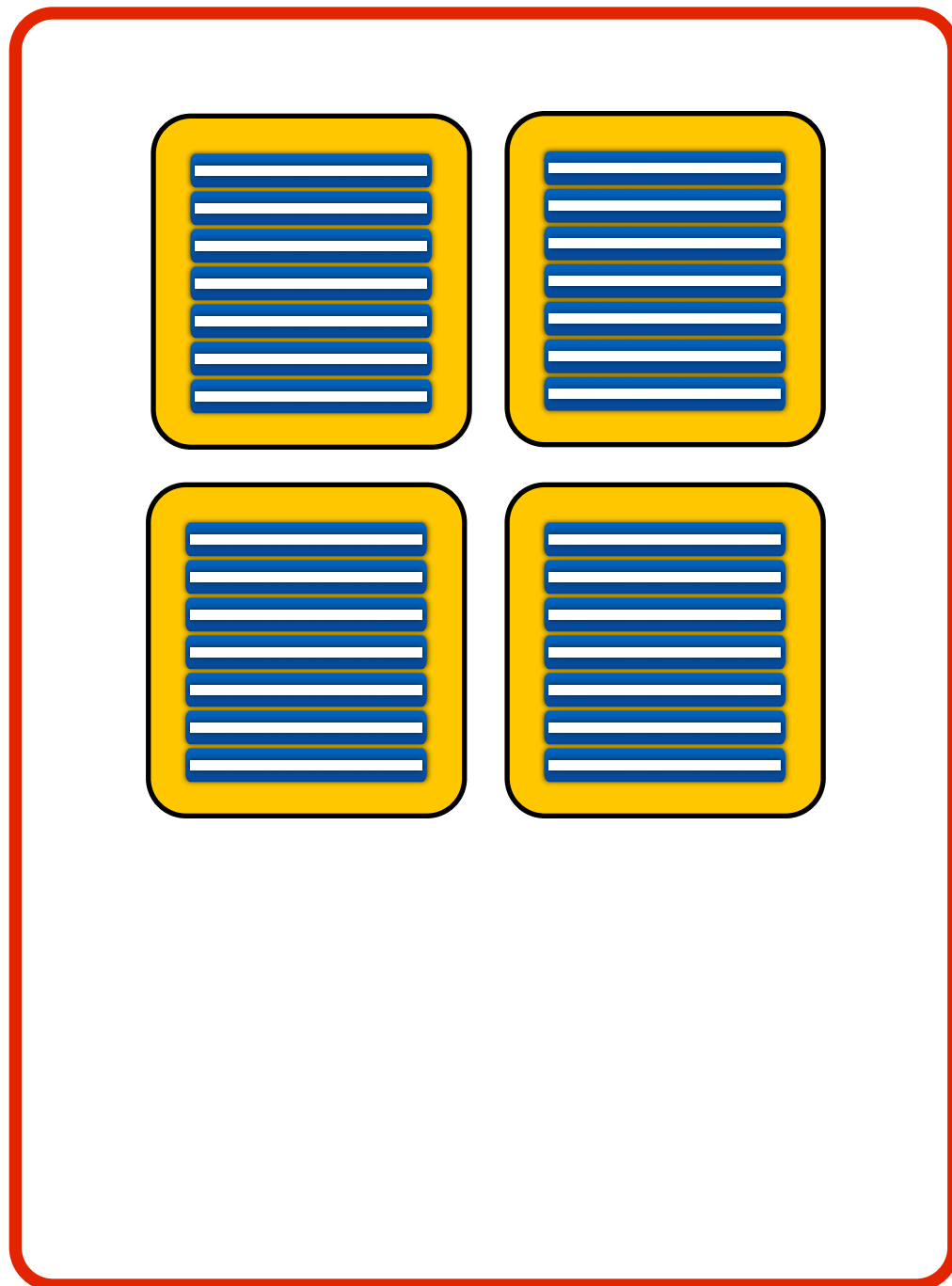


# Generación de “Test Suites”

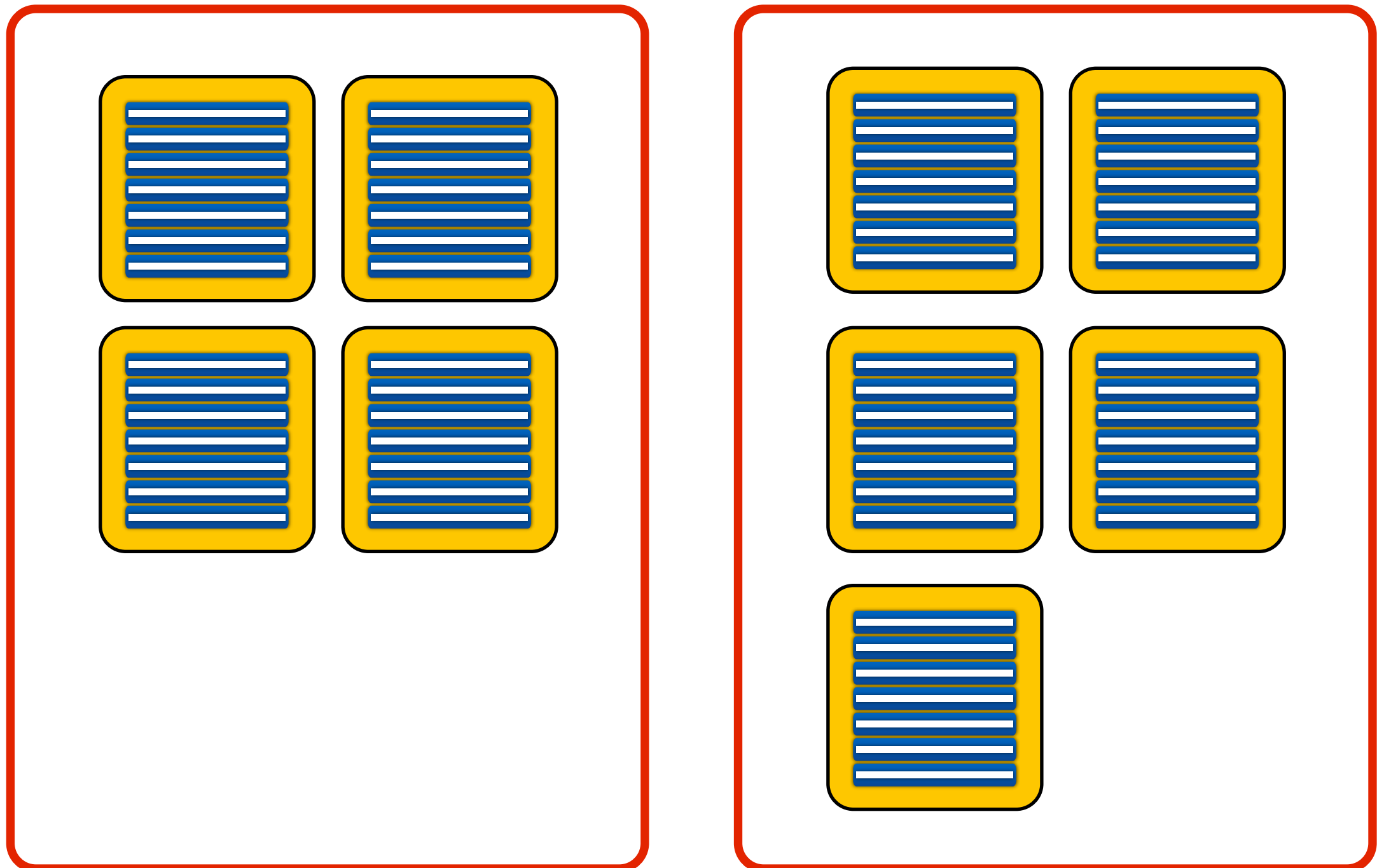




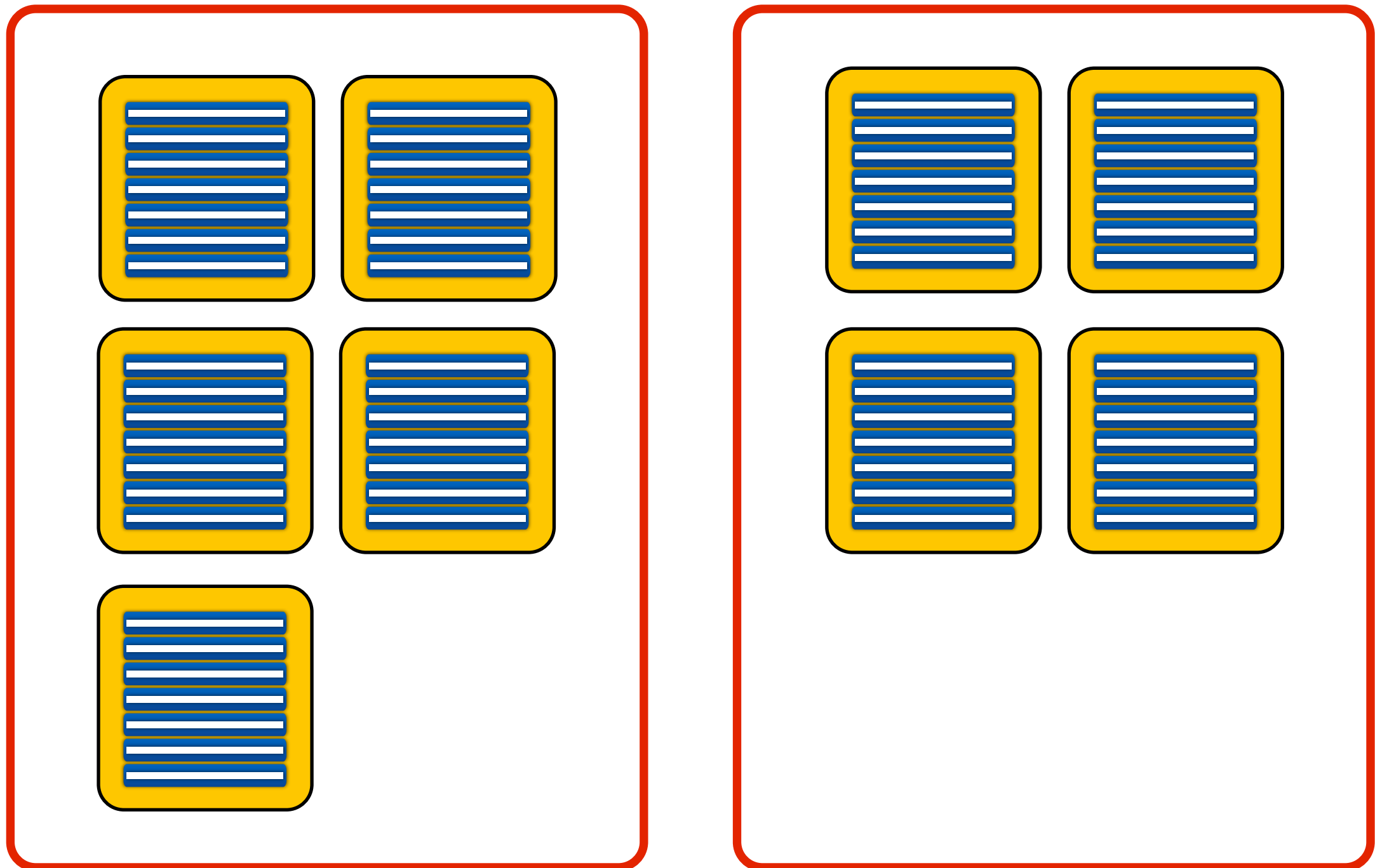
# Generación de “Test Suites”



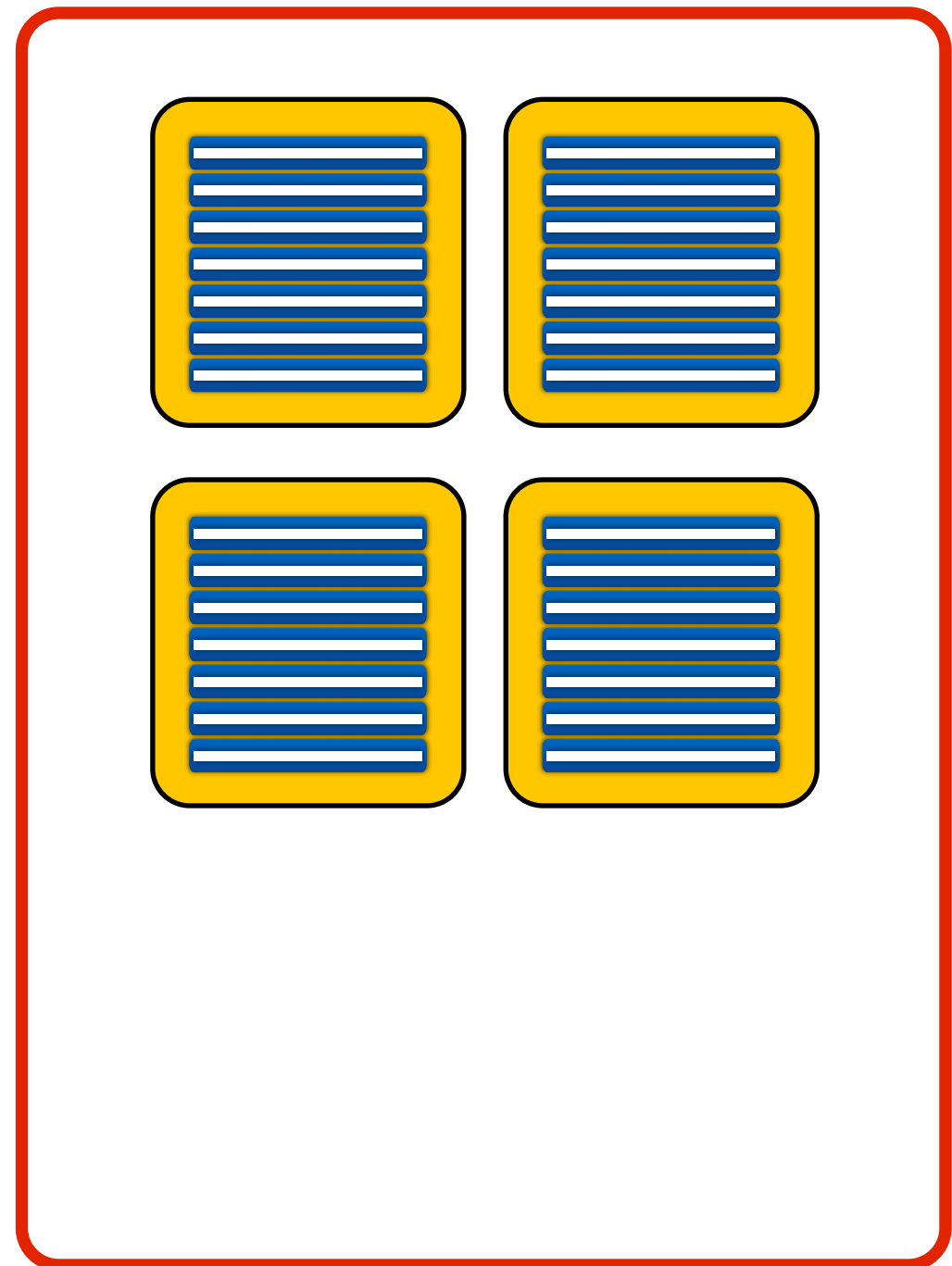
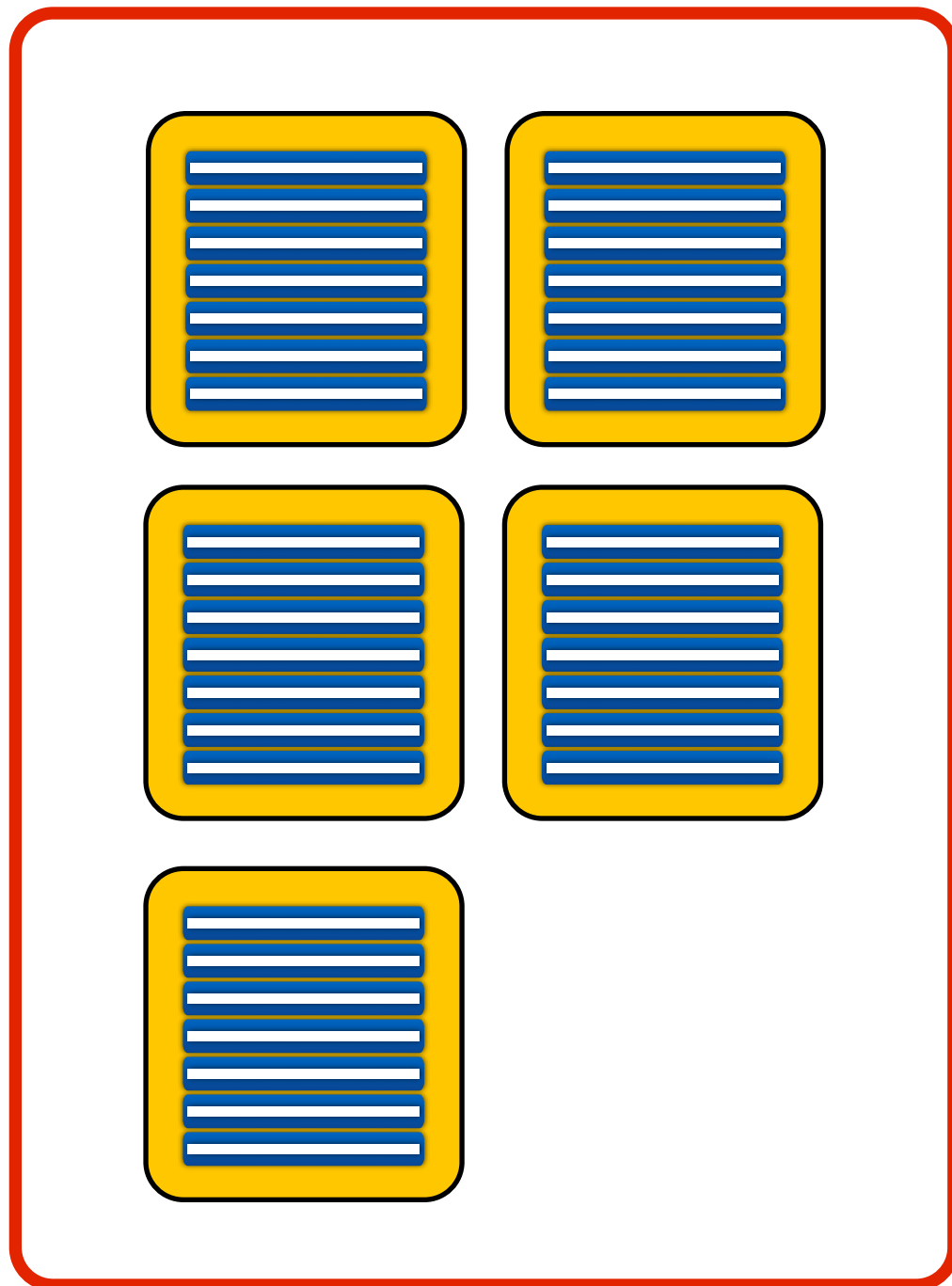
# Generación de “Test Suites”



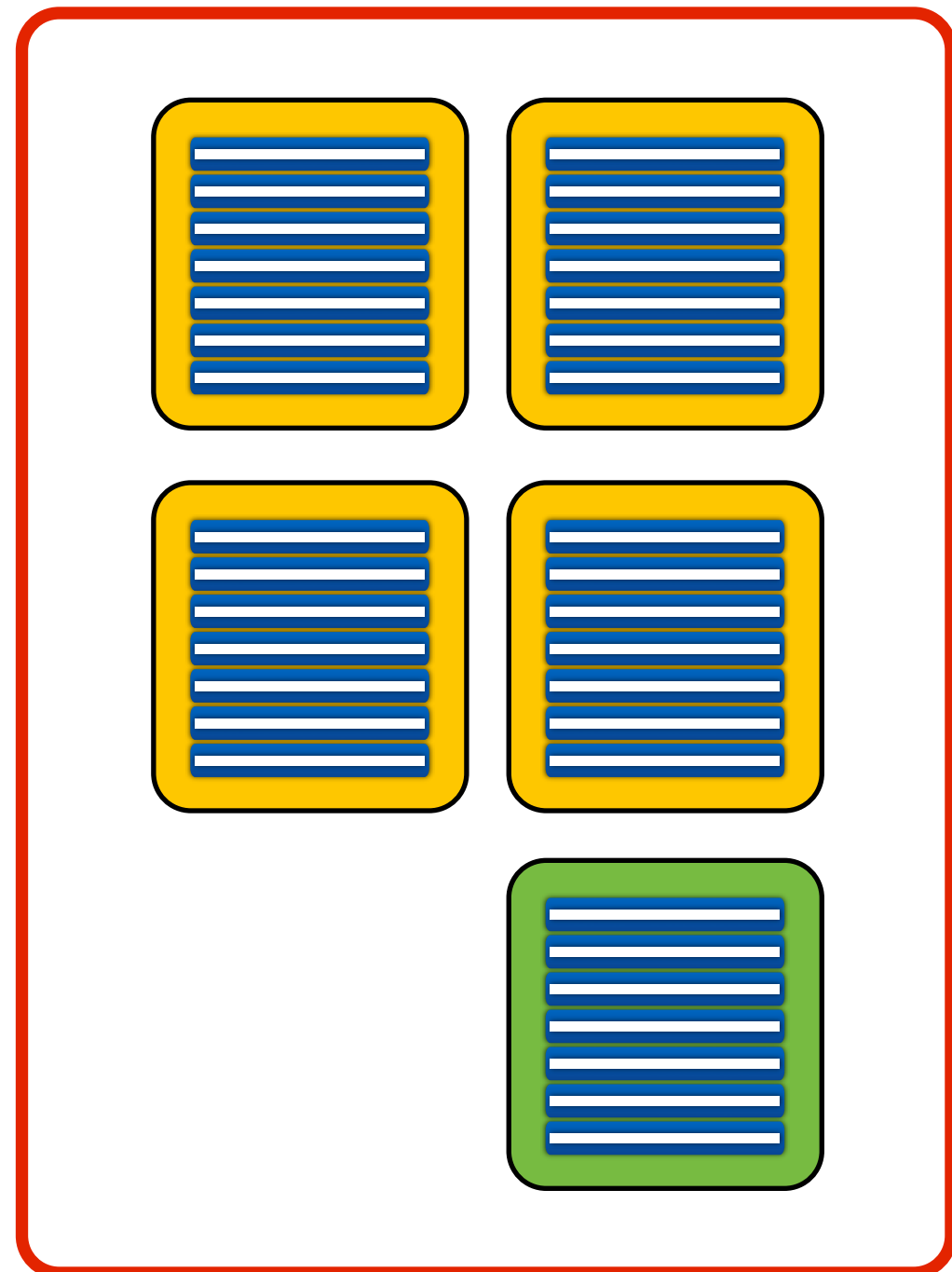
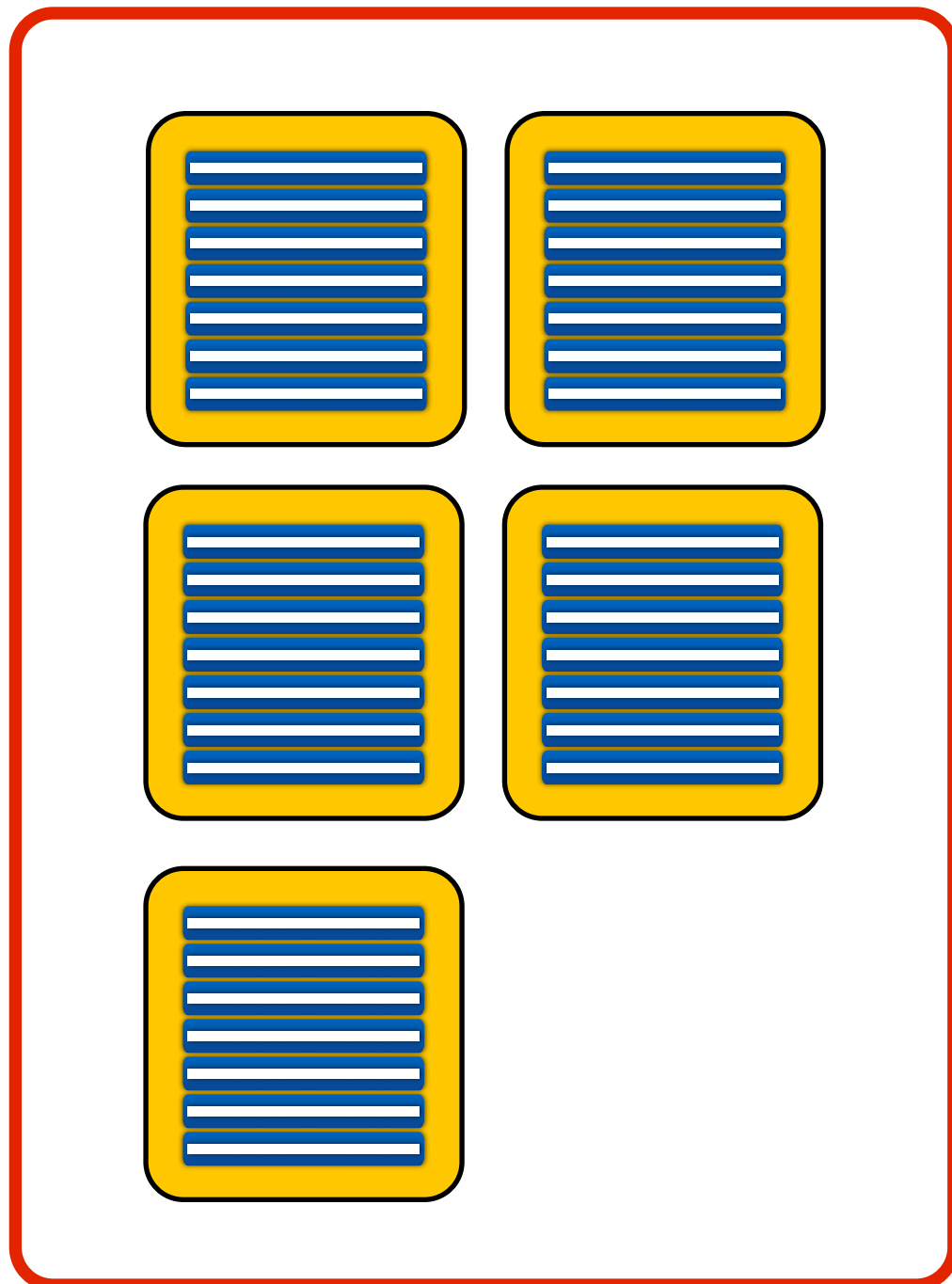
# Cruzamiento (Crossover)



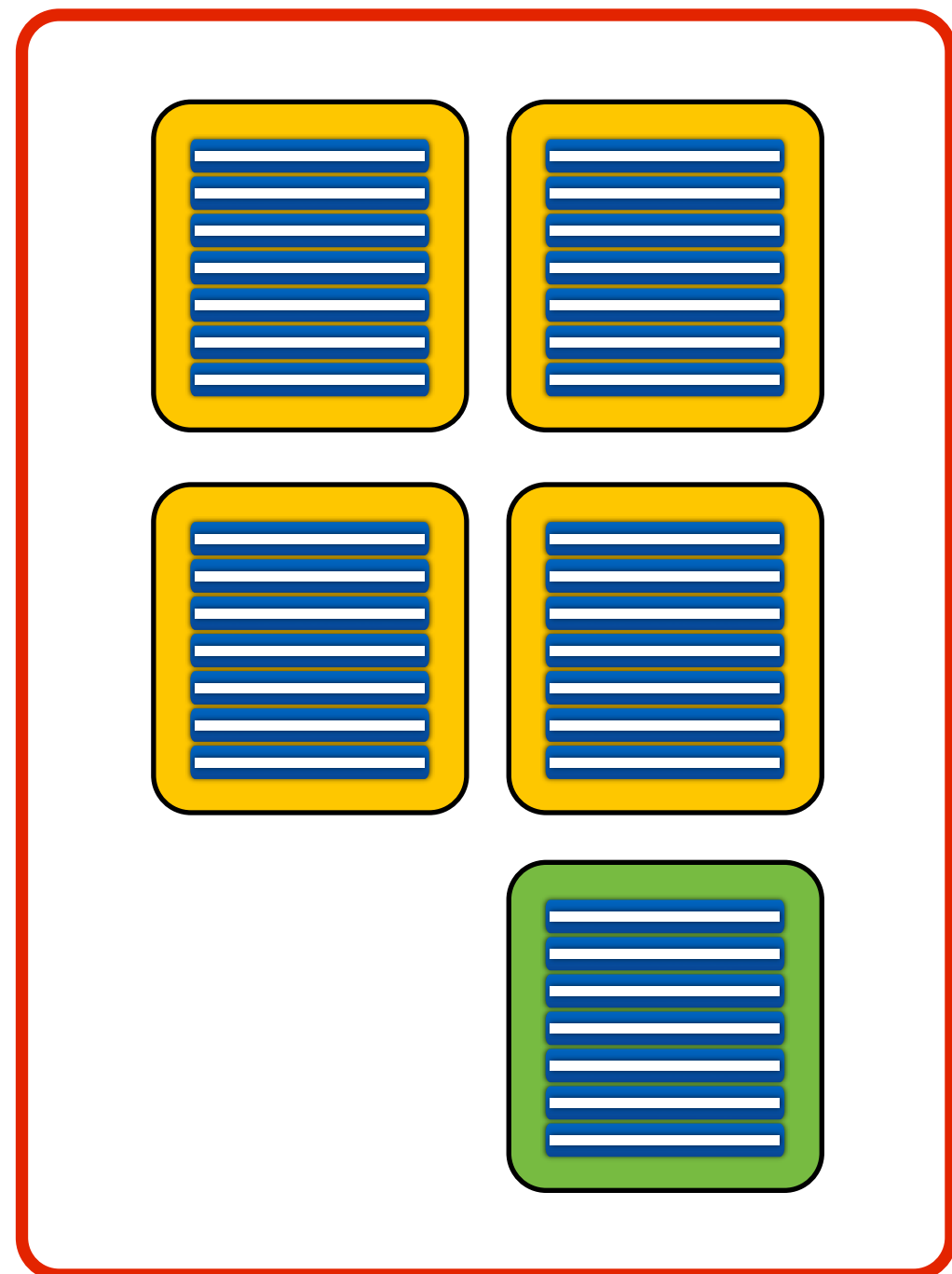
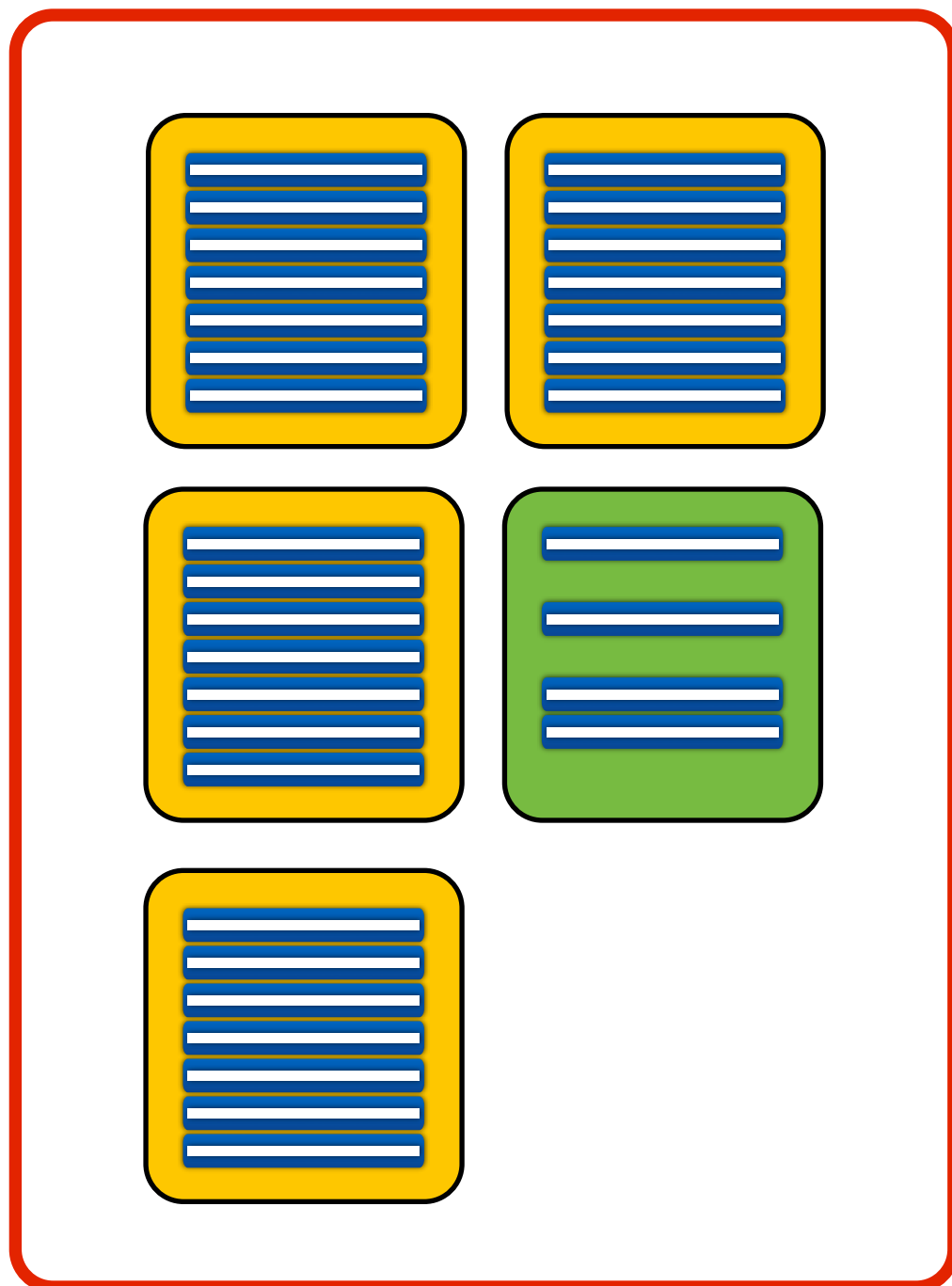
# Mutación



# Mutación

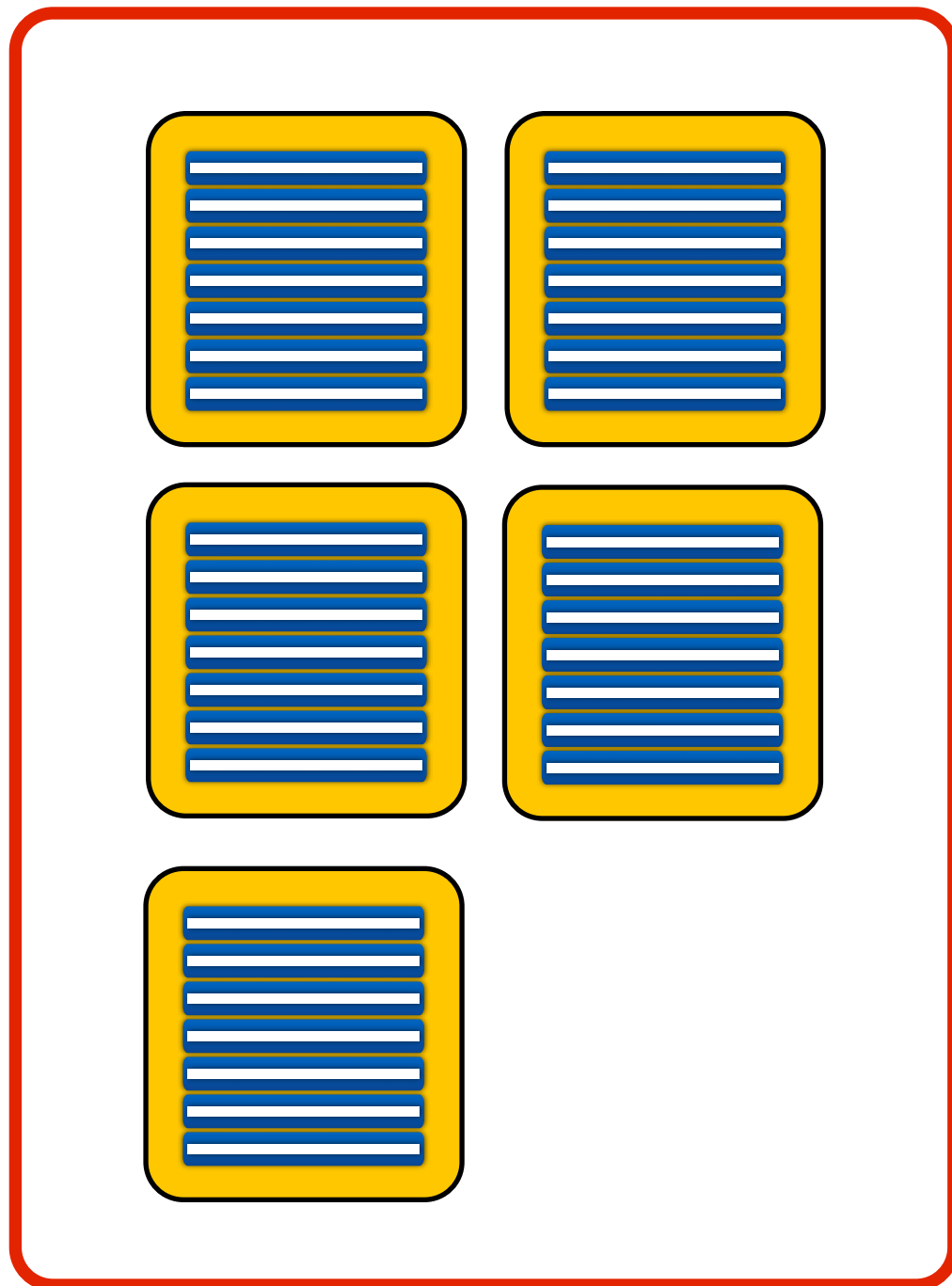


# Mutación



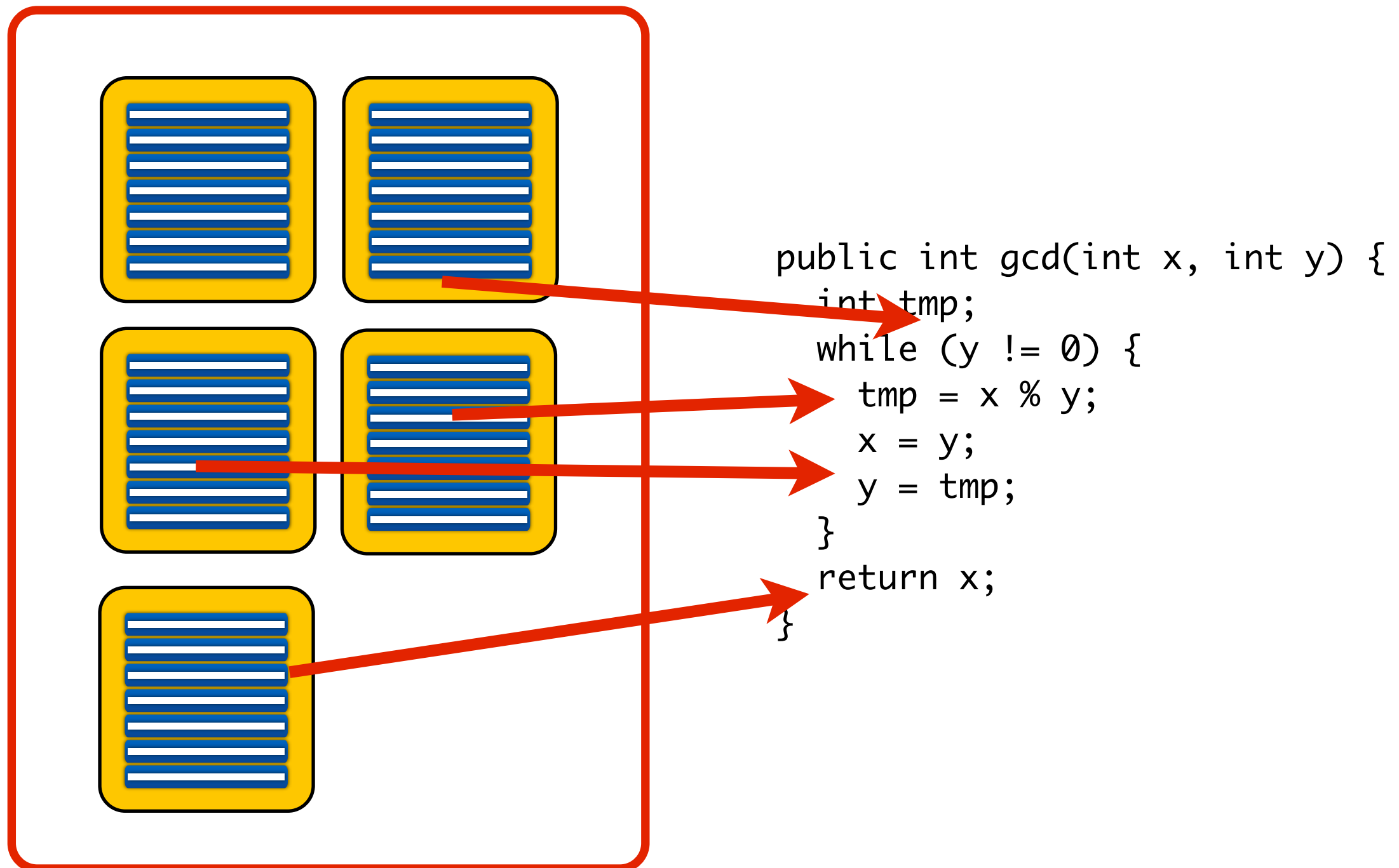


# Fitness



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

# Fitness



# Código Abierto

<http://www.evosite.org/>

<https://github.com/evosite/evosite>

- Paquete Jar – Uso en línea de comandos
- Plug-ins IntelliJ y Eclipse – Uso durante el desarrollo
- Plug-in Maven – Uso automatizado
- Plug-in Jenkins – Uso en integración continua

# Uso en Línea de Comandos

- Opciones principales
  - projectCP
  - class
  - criterion
- Propiedades
  - Search budget (s)
    - Dsearch\_budget=60
  - Generación de aserciones
    - Dassertions=false
    - Dassertion\_strategy=all
  - Minimización (length and values)
    - Dminimize=false
  - Inlining
    - Dinline=false

# DEMO

# Ventajas?

consistency  
corner less tests better design  
present cases stop regression  
**coverage** bug test  
effective cost time  
conferences risk fast speed feedback manual efficiency design future  
failproof framework inspiration to test new cases  
you get to present

## A detailed investigation of the effectiveness of *whole* test suite generation

José Miguel Rojas<sup>1</sup> · Mattia Vivanti<sup>2</sup> ·  
Andrea Arcuri<sup>3,4</sup> · Gordon Fraser<sup>1</sup>

## Code Coverage (EMSE 2017)

## Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges

Sina Shamshiri\*, René Just<sup>†</sup>, José Miguel Rojas\*, Gordon Fraser\*, Phil McMinn\* and Andrea Arcuri<sup>‡</sup>

\*Department of Computer Science, University of Sheffield, UK

<sup>†</sup>Department of Computer Science & Engineering, University of Washington, Seattle, WA, USA

<sup>‡</sup>Scienta, Norway, and University of Luxembourg

\*{sina.shamshiri, j.rojas, gordon.fraser, p.mcminn}@sheffield.ac.uk, <sup>†</sup>rjust@cs.washington.edu, <sup>‡</sup>aa@scienta.no

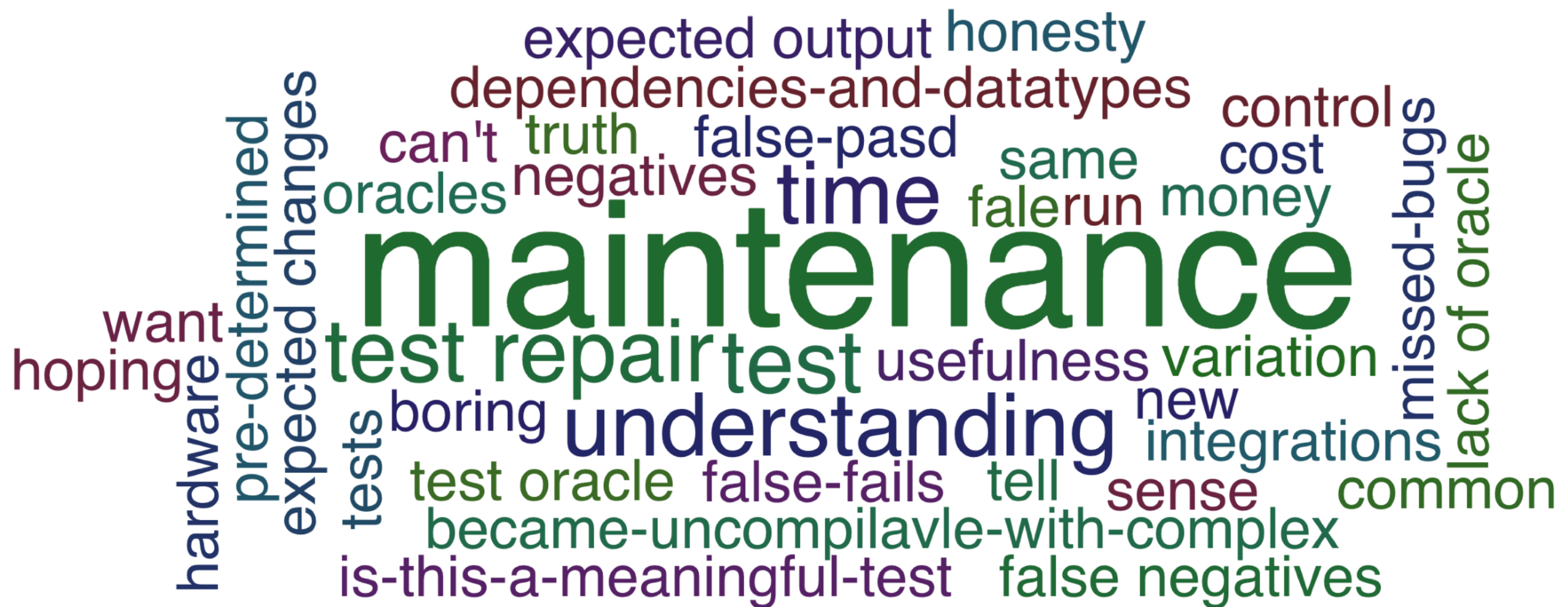
**Abstract**—Rather than tediously writing unit tests manually, tools can be used to generate them automatically — sometimes

even resulting in higher code coverage than manual testing. faults from open source projects [25]. We applied three state-of-the-art unit test generation tools for Java, RANDOOP [30],

## Fault Detection (ASE'15)



# Desventajas?



```
@Test(timeout = 4000)
public void test01() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add((Object) linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch (StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

```
@Test
public void test002() throws Throwable {

    if (debug) { System.out.format("%n%s%n", "RegressionTest2.test002"); }

    java.lang.Object[] obj_array1 = new java.lang.Object[] { (short)10 };
    collections.comparators.FixedOrderComparator fixedOrderComparator2 =
        new collections.comparators.FixedOrderComparator(obj_array1);
    java.util.Comparator comparator3 = fixedOrderComparator2.reversed();
    java.util.Comparator comparator4 = comparator3.reversed();
    java.lang.Object[] obj_array6 = new java.lang.Object[] { (short)10 };
    collections.comparators.FixedOrderComparator fixedOrderComparator7 =
        new collections.comparators.FixedOrderComparator(obj_array6);
    java.util.Comparator comparator8 =
        comparator4.thenComparing((java.util.Comparator)fixedOrderComparator7);
    boolean b10 = fixedOrderComparator7.add((java.lang.Object)10L);
    int i11 = fixedOrderComparator7.getUnknownObjectBehavior();
    boolean b12 = fixedOrderComparator7.isLocked();
    fixedOrderComparator7.setUnknownObjectBehavior( 0 );
    ...
    ...
}
```

```

/**
 * Chromosome :
 1)----->org.apache.commons.lang3.math.NumberUtils[]
 2)----->min[[30823,57]],
 3)----->toString[]
 Covered Branches:[1, 113, 114, 115, 109, 111]
 */
@Test public void TestCase1() throws Throwable {
    NumberUtils clsUTNumberUtils=null;
    clsUTNumberUtils=new NumberUtils();
    short[] clsUTNumberUtilsP2P1=new short[]{30823,57};
    short clsUTNumberUtilsP2R=0;
    clsUTNumberUtilsP2R=NumberUtils.min(clsUTNumberUtilsP2P1);
    assertTrue(Arrays.equals(new short[]{30823,57},clsUTNumberUtilsP2P1));
    assertEquals(57,clsUTNumberUtilsP2R);
    String clsUTNumberUtilsP3R=null;
    clsUTNumberUtilsP3R=clsUTNumberUtils.toString();
    String clsUTNumberUtilsP3RP0P1=new
String("YfJ0ufsFnsYRrFHBpCyLgDWEhIRP0GISMsFFCYjM0SisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLLYPBYmyihySJwWEVETnqz
BRdLRcmk");
    Double clsUTNumberUtilsP3RP0P2P100=-11.825526454314883D;
    Object clsUTNumberUtilsP3RP0P2P1=clsUTNumberUtilsP3RP0P2P100;
    String clsUTNumberUtilsP3RP0P2P200=new String(">?]&#x2D; (;jiah+un");
    Object clsUTNumberUtilsP3RP0P2P2=clsUTNumberUtilsP3RP0P2P200;
    Short clsUTNumberUtilsP3RP0P2P300=1689;
    Object clsUTNumberUtilsP3RP0P2P3=clsUTNumberUtilsP3RP0P2P300;
    Object[] clsUTNumberUtilsP3RP0P2=new Object[]
{clsUTNumberUtilsP3RP0P2P1,clsUTNumberUtilsP3RP0P2P2,clsUTNumberUtilsP3RP0P2P3};
    String clsUTNumberUtilsP3RP0R=null;
    clsUTNumberUtilsP3RP0R=String.format(clsUTNumberUtilsP3RP0P1,(Object[])clsUTNumberUtilsP3RP0P2);

    assertEquals("YfJ0ufsFnsYRrFHBpCyLgDWEhIRP0GISMsFFCYjM0SisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLLYPBYmyihySJwWE
VETnqzBRdLRcmk",clsUTNumberUtilsP3RP0P1.toString());

    assertEquals("YfJ0ufsFnsYRrFHBpCyLgDWEhIRP0GISMsFFCYjM0SisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLLYPBYmyihySJwWE
VETnqzBRdLRcmk",clsUTNumberUtilsP3RP0R);
}

```





# Test Readability

“[Developers] read tests [...] 77% of the total time they spend in them”

Moritz Beller, Georgios Gousios, Annibale Panichella, and Andy Zaidman, “When, How, and Why Developers (Do Not) Test in Their IDEs”. FSE 2015.

# Generación de Casos de Prueba con Nombres Descriptivos

*joint work with Ermira Daka and Gordon Fraser  
presented at ISSTA'17*



# Generación de Casos de Prueba con Nombres Descriptivos



*joint work with Ermira Daka and Gordon Fraser  
presented at ISSTA'17*

```
@Test(timeout = 4000)
public void test1() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add((Object) linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch (StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

```
@Test(timeout = 4000)
public void test1() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add((Object) linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch (StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

# Coverage-Based Naming

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

## Method Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

Method Coverage



# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

testCreateShoppingCart  
testAddPrice  
testGetTotal

## Method Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

## Exception Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

## Exception Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

testAddPriceThrowsIAE

## Exception Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Output Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

Output Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

testGetTotalReturnsZero  
testGetTotalReturnsPositive  
testGetTotalReturnsNegative

Output Coverage



# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Input Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

Input Coverage

# Coverage-Based Naming

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public int getTotal() {  
        return total;  
    }  
}
```

testAddPriceWithZero  
testAddPriceWithPositive  
testAddPriceWithNegative

Input Coverage

# Síntesis de Nombres de Casos de Prueba

|       |       |
|-------|-------|
| Test1 | Test2 |
| Test3 | Test4 |

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*



# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input



# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input



# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input
3. Traducir objetivos a nombres

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input
3. Traducir objetivos a nombres
4. Resolver conflictos: Unicidad entre pares de casos de prueba

# Síntesis de Nombres de Casos de Prueba

## Test1

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningFalse  
I: addPriceWithPositive

## Test2

M: createsShoppingCart  
M: addPrice  
E: addPriceThrowsIAE  
I: addPriceWithZero

## Test3

M: createsShoppingCart  
M: addPrice  
O: addPriceReturningTrue  
I: addPriceWithPositive

## Test4

M: createsShoppingCart  
M: getTotal  
O: getTotalReturningZero

1. Identificar objetivos cubiertos  
*únicamente por cada caso de prueba*
2. Ordenar objetivos cubiertos  
Exception > Method > Output > Input
3. Traducir objetivos a nombres
4. Resolver conflictos: Unicidad entre pares de casos de prueba
5. Sufijos numéricos

# DEMO

# Evaluación Empírica

# Evaluación Empírica

RQ1: Do developers agree with synthesised test names?

# Evaluación Empírica

RQ1: Do developers agree with synthesised test names?

RQ2: Can developers match tests with synthesised test names?



# Evaluación Empírica

RQ1: Do developers agree with synthesised test names?

RQ2: Can developers match tests with synthesised test names?

RQ3: Can synthesised test names help developers to match tests to code under test?



# Evaluación Empírica

# Evaluación Empírica



Estudio en línea

# Evaluación Empírica



Estudio en línea



47 participantes

# Evaluación Empírica



Estudio en línea



47 participantes



10  
clases Java

# Evaluación Empírica



Estudio en línea



47 participantes



10  
clases Java



2 expertos  
como control

# Evaluación Empírica



Estudio en línea



47 participantes



10  
clases Java



2 expertos  
como control



Sin tiempo límite

# Test Names

50% Complete

## Question 6

For the following test, indicate your level of agreement with the suggested test name "testDigitWildcardTakingCharacter".

Test

Same test in context

```
@Test
public void testDigitWildcardTakingCharacter() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#@}:*Q54)M!");
    Character character0 = Character.valueOf(':');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#@}:*Q54)M!");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

**testDigitWildcardTakingCharacter** is an appropriate name for this test.

|                                                              |                                                            |                                                 |                                                         |                                                          |
|--------------------------------------------------------------|------------------------------------------------------------|-------------------------------------------------|---------------------------------------------------------|----------------------------------------------------------|
| <input type="radio"/>                                        | <input type="radio"/>                                      | <input type="radio"/>                           | <input type="radio"/>                                   | <input type="radio"/>                                    |
| <b>Strongly disagree</b>                                     | <b>Disagree</b>                                            | <b>Neutral</b>                                  | <b>Agree</b>                                            | <b>Strongly agree</b>                                    |
| This test name is completely inappropriate and unresponsive. | This test name is somewhat inappropriate and unresponsive. | Neither agree nor disagree with this test name. | This test name is somewhat appropriate and descriptive. | The test name is completely appropriate and descriptive. |

**Why? Please explain your answer here:**

For example: "That's exactly how I would have named the test", "It's not entirely clear by the name what method is being tested", "The name is unnecessarily long".

« Previous

Next »



# Test Names

50% Complete

Question 6

For the following test, indicate your level of agreement with the suggested test name "testDigitWildcardTakingCharacter".

Test

Same test in context

```
@Test
public void testDigitWildcardTakingCharacter() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#*:Q54)M!");
    Character character0 = Character.valueOf(' ');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#*:Q54)M!");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

testDigitWildcardTakingCharacter is an appropriate name for this test.

Strongly disagree

Disagree

Neutral

Agree

Strongly agree

This test name is completely inappropriate and undescriptive.

This test name is somewhat inappropriate and undescriptive.

Neither agree nor disagree with this test name.

This test name is somewhat appropriate and descriptive.

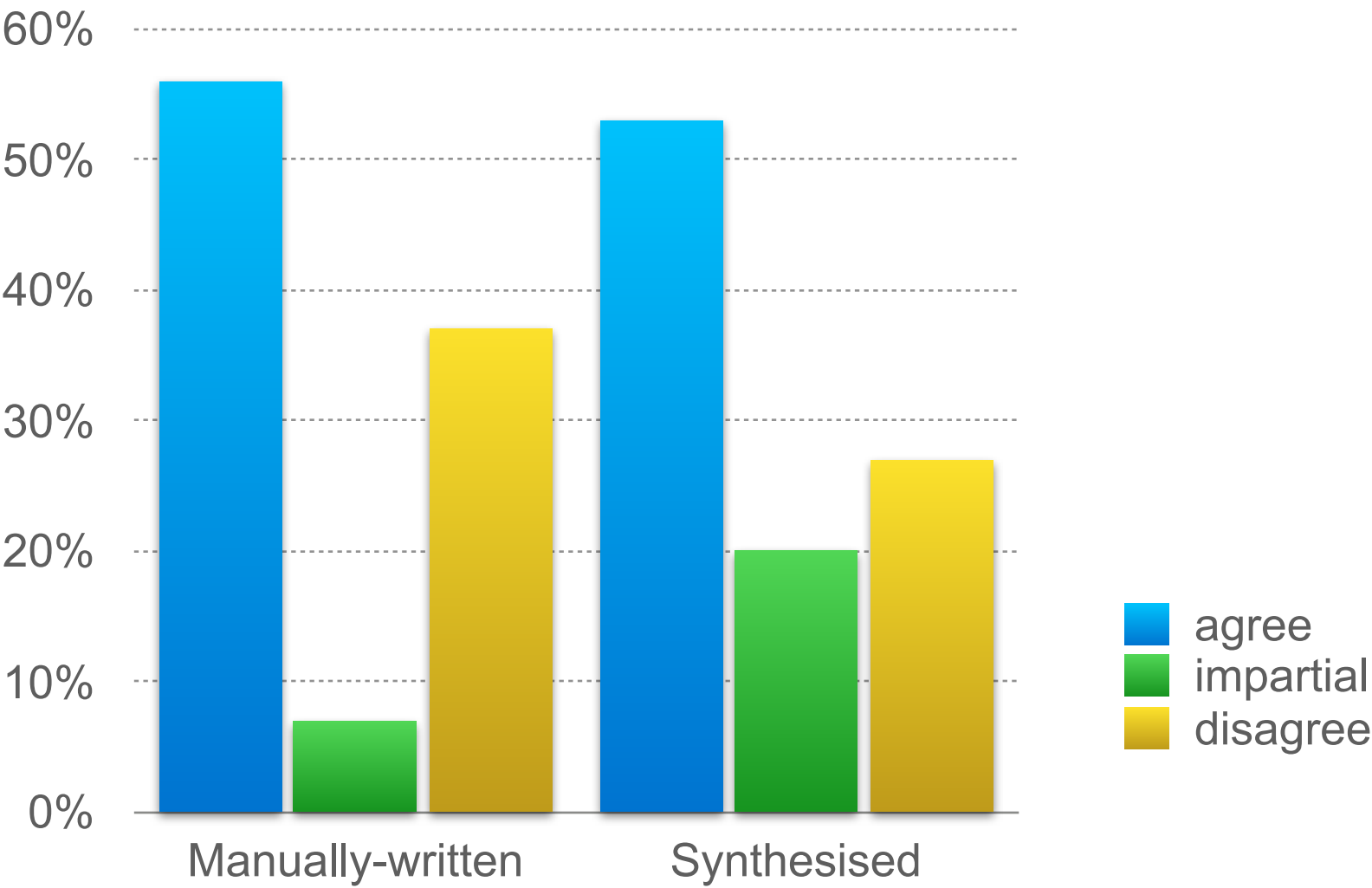
The test name is completely appropriate and descriptive.

Why? Please explain your answer here:

For example: "That's exactly how I would have named the test", "It's not entirely clear by the name what method is being tested", "The name is unnecessarily long".

« Previous

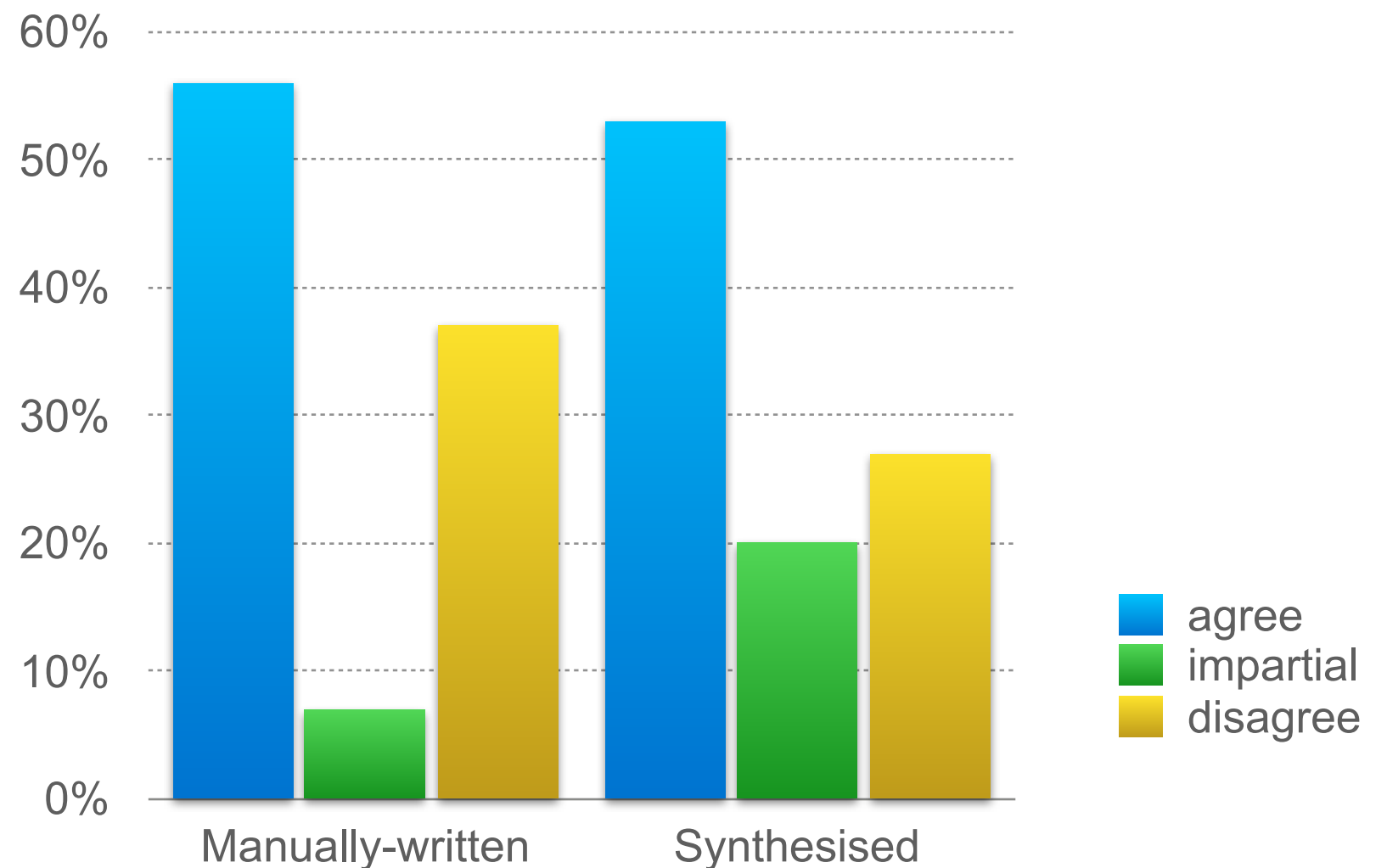
Next »





# Test Names

“Do you agree with the name of the following test?”



50% Complete

Question 6

For the following test, indicate your level of agreement with the suggested test name "testDigitWildcardTakingCharacter".

Test Same test in context

```
@Test
public void testDigitWildcardTakingCharacter() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#0:*Q54)M!");
    Character character0 = Character.valueOf(' ');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#0:*Q54)M!");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

testDigitWildcardTakingCharacter is an appropriate name for this test.

**Strongly disagree** This test name is completely inappropriate and undescriptive.

**Disagree** This test name is somewhat inappropriate and undescriptive.

**Neutral** Neither agree nor disagree with this test name.

**Agree** This test name is somewhat appropriate and descriptive.

**Strongly agree** The test name is completely appropriate and descriptive.

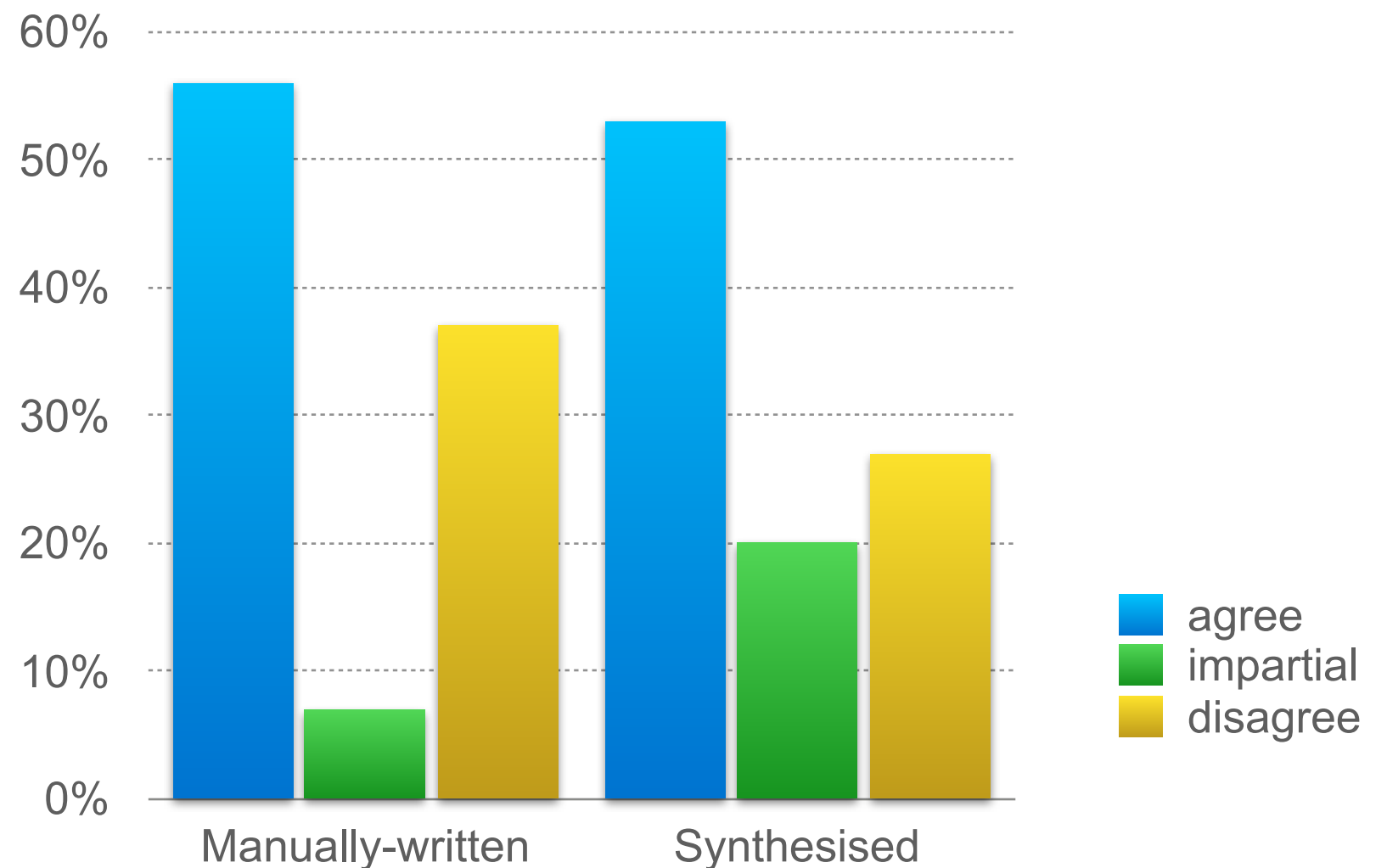
Why? Please explain your answer here:

For example: "That's exactly how I would have named the test", "It's not entirely clear by the name what method is being tested", "The name is unnecessarily long".

« Previous Next »

# Test Names

“Do you agree with the name of the following test?”



50% Complete

Question 6

For the following test, indicate your level of agreement with the suggested test name "testDigitWildcardTakingCharacter".

Test Same test in context

```
@Test
public void testDigitWildcardTakingCharacter() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#0:*Q54)M1");
    Character character0 = Character.valueOf(' ');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#0:*Q54)M1");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

testDigitWildcardTakingCharacter is an appropriate name for this test.

**Strongly disagree** This test name is completely inappropriate and undescriptive.

**Disagree** This test name is somewhat inappropriate and undescriptive.

**Neutral** Neither agree nor disagree with this test name.

**Agree** This test name is somewhat appropriate and descriptive.

**Strongly agree** The test name is completely appropriate and descriptive.

Why? Please explain your answer here:

For example: "That's exactly how I would have named the test", "It's not entirely clear by the name what method is being tested", "The name is unnecessarily long".

« Previous Next »

Developers agreed similarly—and disagreed less—with synthesized test names than with manually given names.

# Test Names and Test Code

40% Complete

## Question 5

For the following test, select the test name that you think is most appropriate and best describes the test code.

Test

Same test in context

```
@Test
2 public void _____() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#0:*Q54)M!");
    Character character0 = Character.valueOf(':');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#0:*Q54)M!");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

Select a name for the highlighted test above:

- ☐ testNonMatchingPattern
- ☐ testTwoArgConstructorWildcard
- ☐ testSingleArgConstructor
- ☐ None of the above.

Why? Please explain your selection here:

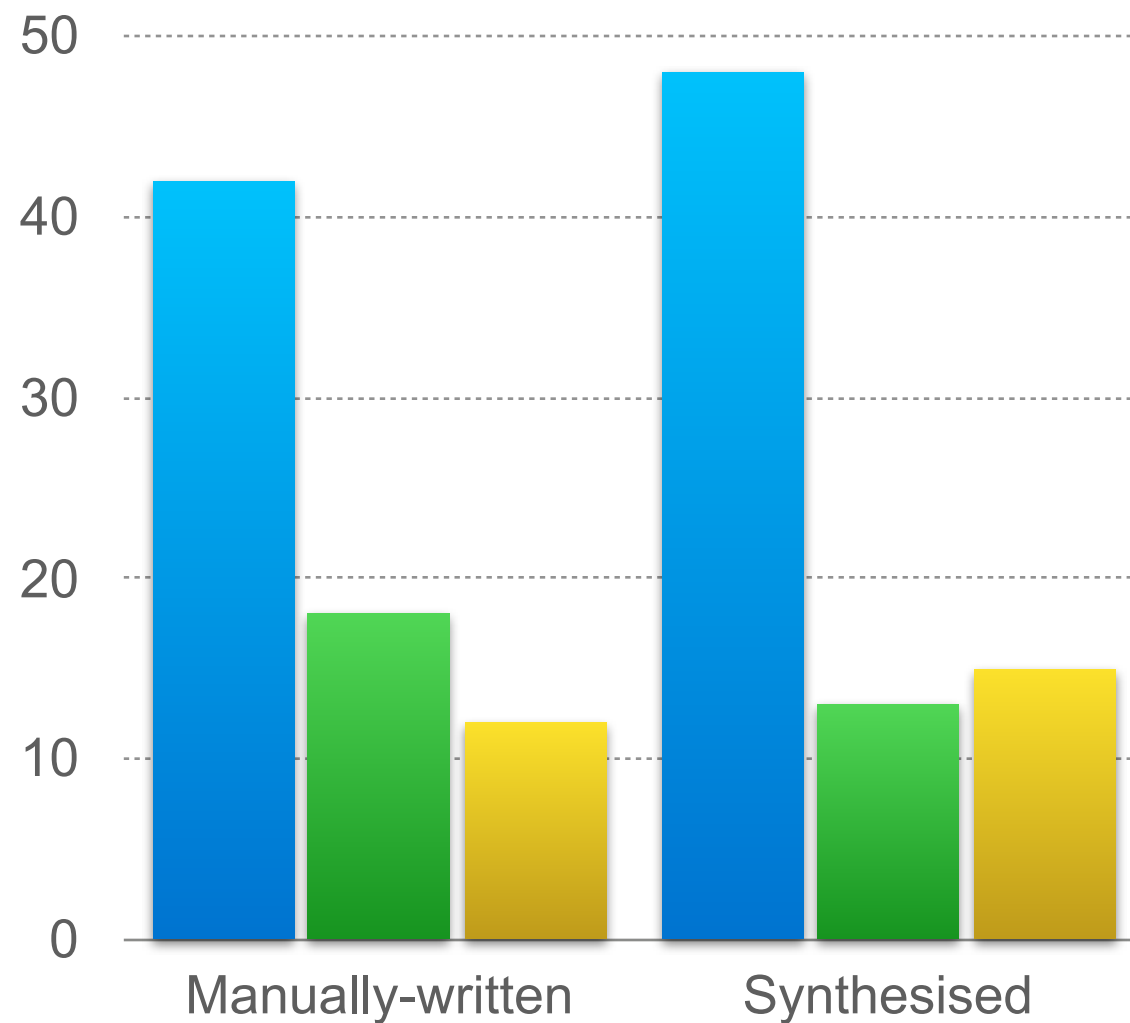
For example: "That's exactly how I would have named the test", "It's the most descriptive name of the three", "The other names are just terrible".

« Previous

Next »

# Test Names and Test Code

“For this test, select the most descriptive name from the list.”



correct  
don't know  
incorrect

40% Complete

### Question 5

For the following test, select the test name that you think is most appropriate and best describes the test code.

Test [Same test in context](#)

```
@Test
public void _____() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#0:*Q54)M1");
    Character character0 = Character.valueOf(':');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#0:*Q54)M1");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

Select a name for the highlighted test above:

- ☐ testNonMatchingPattern
- ☐ testTwoArgConstructorWildcard
- ☐ testSingleArgConstructor
- ☐ None of the above.

Why? Please explain your selection here:

For example: "That's exactly how I would have named the test", "It's the most descriptive name of the three", "The other names are just terrible".

[« Previous](#) [Next »](#)

# Test Names and Test Code

“For this test, select the most descriptive name from the list.”

40% Complete

### Question 5

For the following test, select the test name that you think is most appropriate and best describes the test code.

Test

Same test in context

```
@Test
public void _____() throws Throwable {
    StringPattern stringPattern0 = new StringPattern("2*#*:Q54)M!");
    Character character0 = Character.valueOf(':');
    stringPattern0.digitWildcard(character0);
    boolean boolean0 = stringPattern0.matches("2*#*:Q54)M!");
    assertFalse(boolean0);
    assertFalse(stringPattern0.getIgnoreCase());
}
```

Select a name for the highlighted test above:

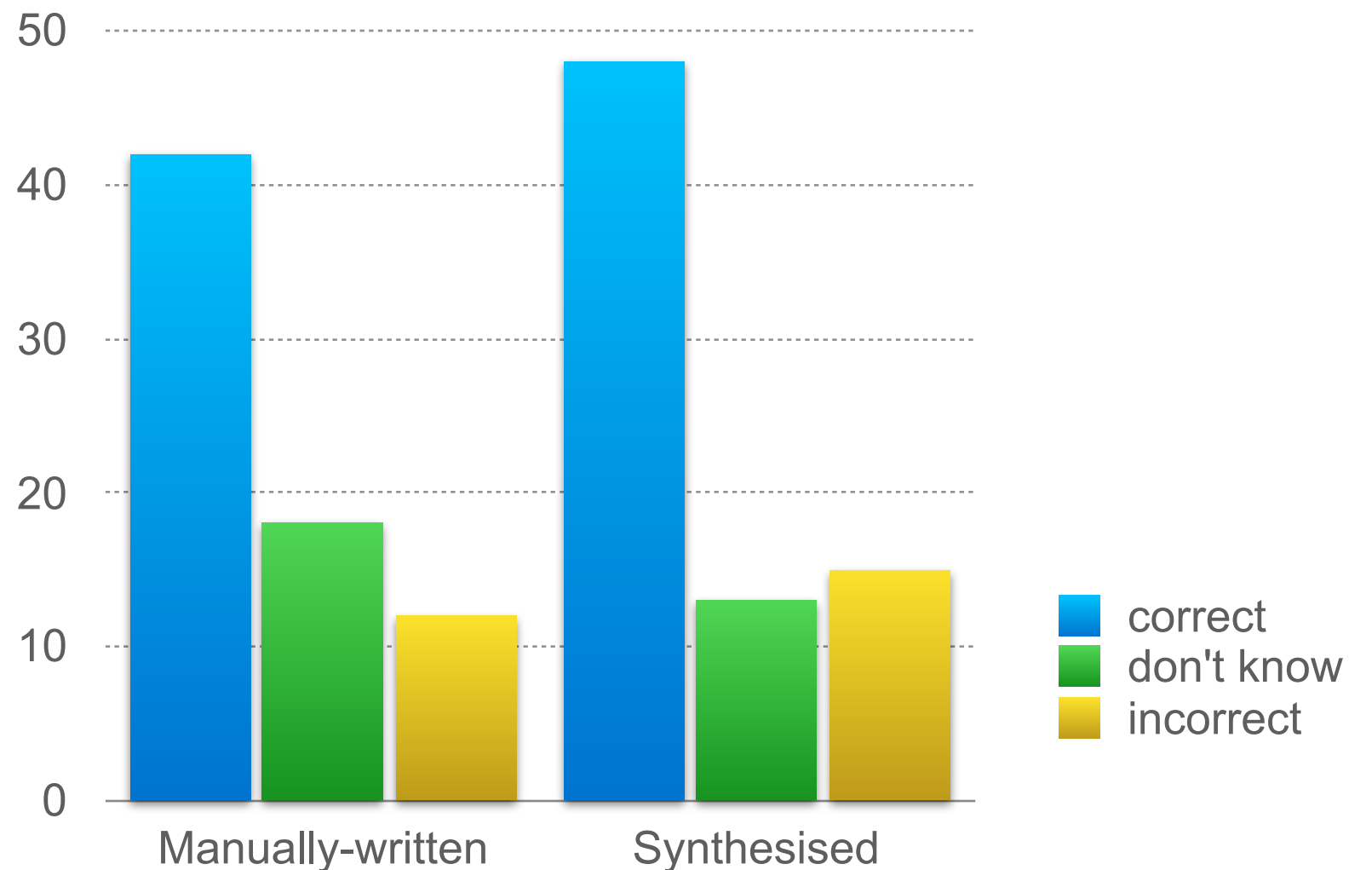
- ☐ testNonMatchingPattern
- ☐ testTwoArgConstructorWildcard
- ☐ testSingleArgConstructor
- ☐ None of the above.

Why? Please explain your selection here:

For example: "That's exactly how I would have named the test", "It's the most descriptive name of the three", "The other names are just terrible".

« Previous

Next »



Developers were slightly more accurate and faster at matching tests with synthesized names.

# Test Names and Code under Test

90% Complete

## Question 4

If a change was made in line number 189 in the following class, which of the tests listed in the right panel would you inspect/run to test the changed behaviour?

Class Under Test

```
173         SimpleRole role = new SimpleRole(roleName);
174         add(role);
175     }
176 }
177
178 public void add(SimpleRole role) {
179     Set<SimpleRole> roles = getSimpleRoles();
180     if (roles == null) {
181         roles = new LinkedHashSet<SimpleRole>();
182         setSimpleRoles(roles);
183     }
184     roles.add(role);
185 }
186
187 public void addRoles(Set<String> roleNames) {
188     if (roleNames != null && !roleNames.isEmpty()) {
189         for (String name : roleNames) {
190             addRole(name);
191         }
192     }
193 }
```

Please select one test:

- ☐ testMergeRoles
- ☐ testEmptySimpleRoles
- ☐ testNullSimpleRoles
- ☒ None of the above.

**Why? Please explain your selection here:**

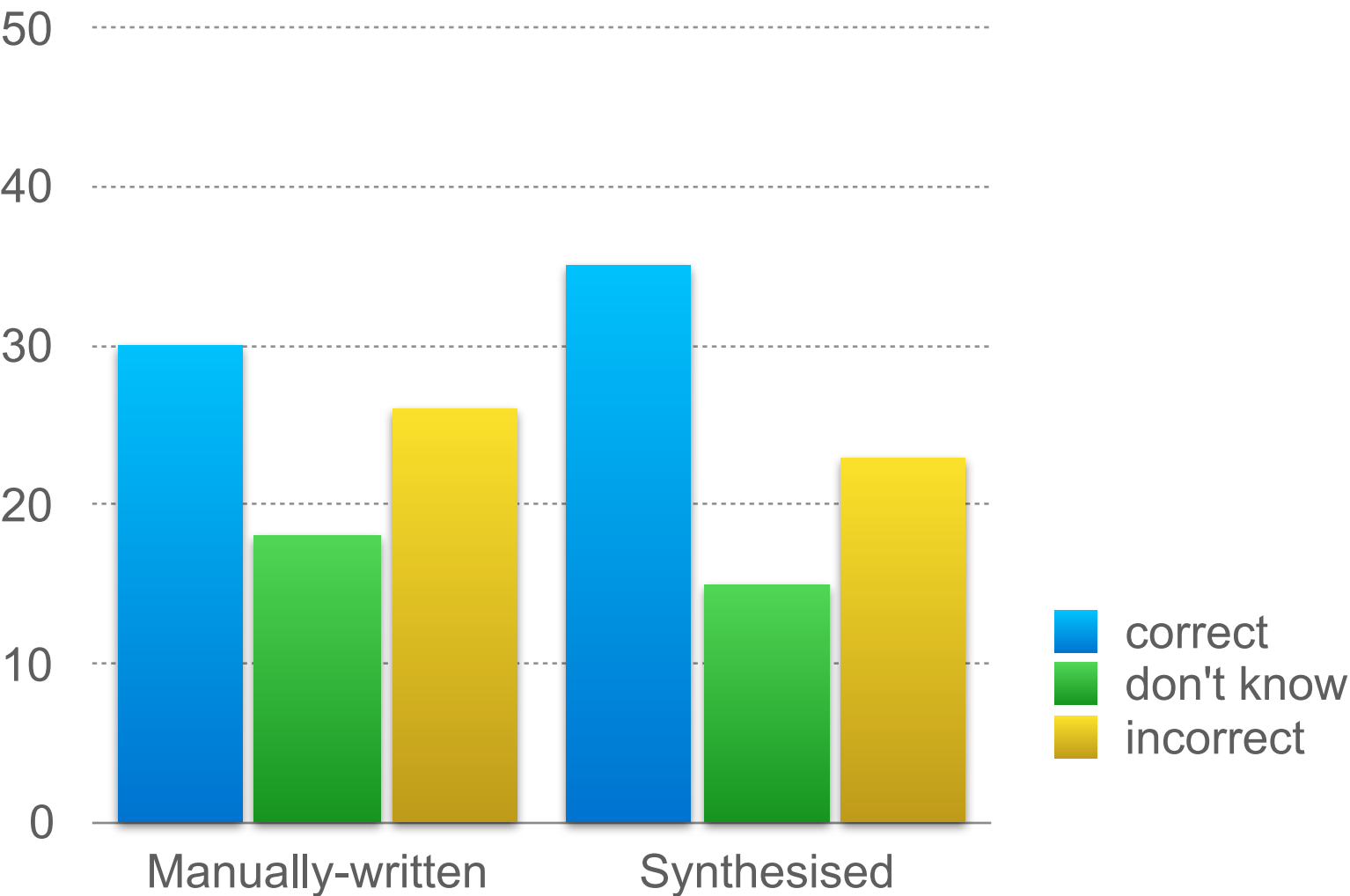
For example: "The name suggests the test will cover that line", "I honestly tried, but I don't know."

« Previous

Next »

# Test Names and Code under Test

“There is a bug in line X, which test would you choose?”



90% Complete

Question 4

If a change was made in line number 189 in the following class, which of the tests listed in the right panel would you inspect/run to test the changed behaviour?

Class Under Test

```
173     SimpleRole role = new SimpleRole(roleName);
174     add(role);
175 }
176
177
178 public void add(SimpleRole role) {
179     Set<SimpleRole> roles = getSimpleRoles();
180     if (roles == null) {
181         roles = new LinkedHashSet<SimpleRole>();
182         setSimpleRoles(roles);
183     }
184     roles.add(role);
185 }
186
187 public void addRoles(Set<String> roleNames) {
188     if (roleNames != null && !roleNames.isEmpty()) {
189         for (String name : roleNames) {
190             addRole(name);
191         }
192     }
193 }
```

Please select one test:

☐ testMergeRoles

☐ testEmptySimpleRoles

☐ testNullSimpleRoles

☒ None of the above.

Why? Please explain your selection here:

For example: "The name suggests the test will cover that line", "I honestly tried, but I don't know."

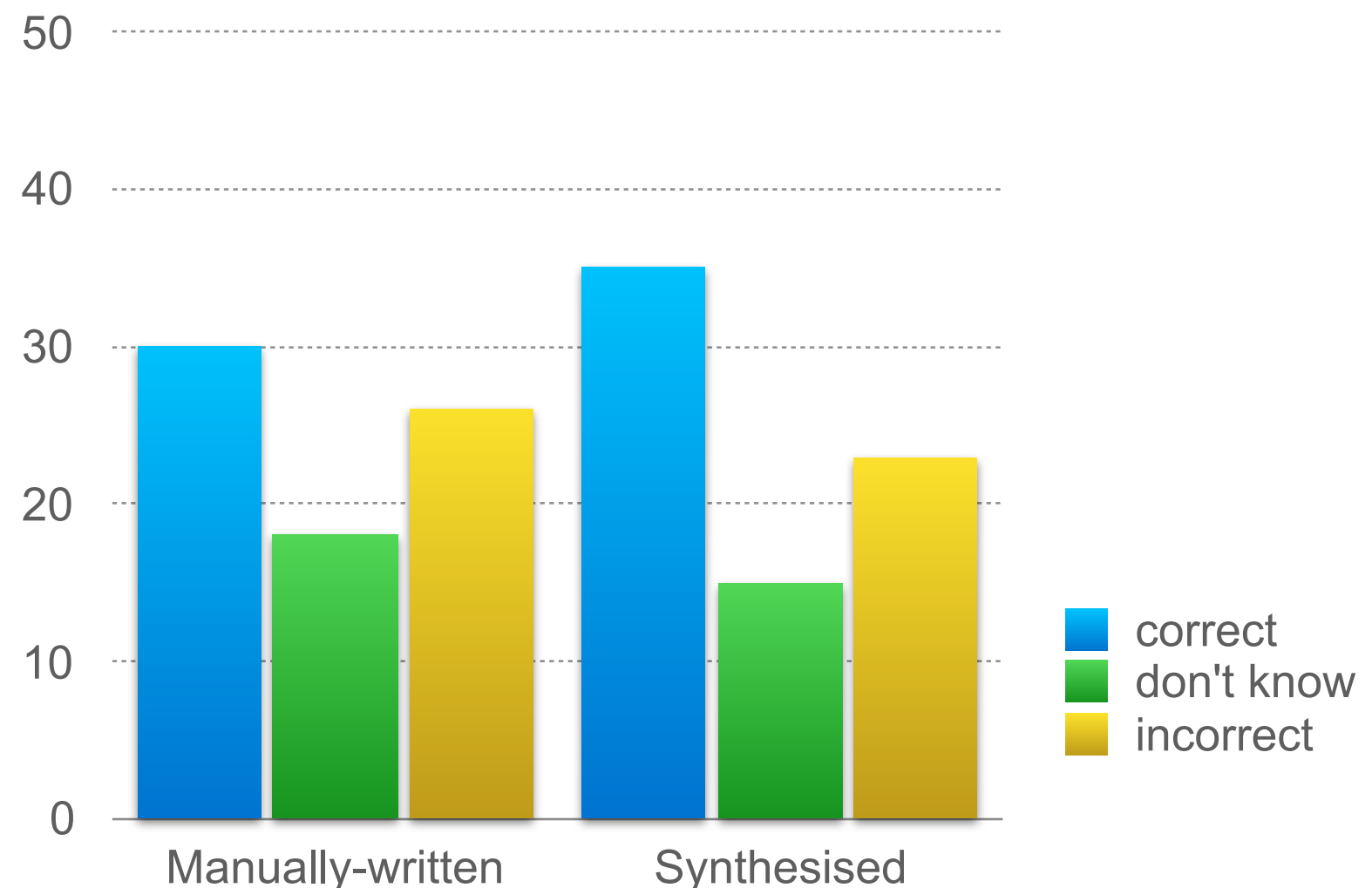
« Previous

Next »



# Test Names and Code under Test

“There is a bug in line X, which test would you choose?”



Developers were more accurate at identifying relevant tests for given pieces of code using synthesised test names.

90% Complete

### Question 4

If a change was made in line number 189 in the following class, which of the tests listed in the right panel would you inspect/run to test the changed behaviour?

Class Under Test

```
173     SimpleRole role = new SimpleRole(roleName);
174     add(role);
175 }
176
177
178 public void add(SimpleRole role) {
179     Set<SimpleRole> roles = getSimpleRoles();
180     if (roles == null) {
181         roles = new LinkedHashSet<SimpleRole>();
182         setSimpleRoles(roles);
183     }
184     roles.add(role);
185 }
186
187 public void addRoles(Set<String> roleNames) {
188     if (roleNames != null && !roleNames.isEmpty()) {
189         for (String name : roleNames) {
190             addRole(name);
191         }
192     }
193 }
```

Please select one test:

☐ testMergeRoles

☐ testEmptySimpleRoles

☐ testNullSimpleRoles

☒ None of the above.

Why? Please explain your selection here:

For example: "The name suggests the test will cover that line", "I honestly tried, but I don't know."

« Previous      Next »



# Tutoriales

- Usar EvoSuite en la línea de comandos:  
<http://www.evosuite.org/documentation/tutorial-part-1/>
- Usar EvoSuite con Maven:  
<http://www.evosuite.org/documentation/tutorial-part-2/>
- Experimentos con EvoSuite:  
<http://www.evosuite.org/documentation/tutorial-part-3/>
- Extender EvoSuite:  
<http://www.evosuite.org/documentation/tutorial-part-4/>

# Trabajo Relacionado

## Software Engineering Gamification, e.g., Code Defenders

The screenshot shows the Code Defenders web application. The browser address bar displays `code-defenders.org/multiplayer/play?id=2260`. The page header includes the Code Defenders logo and navigation links: games, upload class, leaderboard, help, and a user profile for 'nelliwalkinshaw'. The main content area is titled 'DEFENDER::ACTIVE' and 'SparseIntArray'. It features a 'Class Under Test' section with Java code for the `SparseIntArray` class, a 'Write a new JUnit test here' section with a template JUnit test, and a 'JUnit tests' section showing a specific test case. Below the code sections is a table of 'Existing Mutants' with columns for 'alive (35)', 'killed (58)', and 'equivalent (5)'. The table lists four mutants with their creators and UID, and a 'Claim Equivalent' button for each. The bottom of the page has a pagination bar with 'First', 'Previous', '1', '2', '3', '4', '5', '...', '9', 'Next', and 'Last'.

Code Defenders

games upload class leaderboard help nelliwalkinshaw

DEFENDER::ACTIVE SparseIntArray Show Scoreboard

Class Under Test

```
36 /**
37  * Creates a new SparseIntArray containing no mappings that will not
38  * require any additional memory allocation to store the specified
39  * number of mappings. If you supply an initial capacity of 0, the
40  * sparse array will be initialized with a light-weight representation
41  * not requiring any additional array allocations.
42  */
43 public SparseIntArray(int initialCapacity) {
44     if (initialCapacity == 0) {
45         mKeys = SparseIntArray.EMPTY_INT_ARRAY;
46         mValues = SparseIntArray.EMPTY_INT_ARRAY;
47     } else {
48         mKeys = new int[initialCapacity];
49         mValues = new int[mKeys.length];
50     }
51     mSize = 0;
52 }
53 /**
54  * Given the current size of an array, returns an ideal size to which the array should grow
55  * This is typically double the given size, but should not be relied upon to do so in the
56  * future.
57  */
58 public static int growSize(int currentSize) {
59     return currentSize <= 4 ? 8 : currentSize * (currentSize >> 1);
60 }
61
```

Write a new JUnit test here

```
1 /** no package name */
2
3 import org.junit.*;
4 import static org.junit.Assert.*;
5
6 public class TestSparseIntArray {
7     @Test(timeout = 4000)
8     public void test() throws Throwable {
9         // test here!
10     }
11 }
12
```

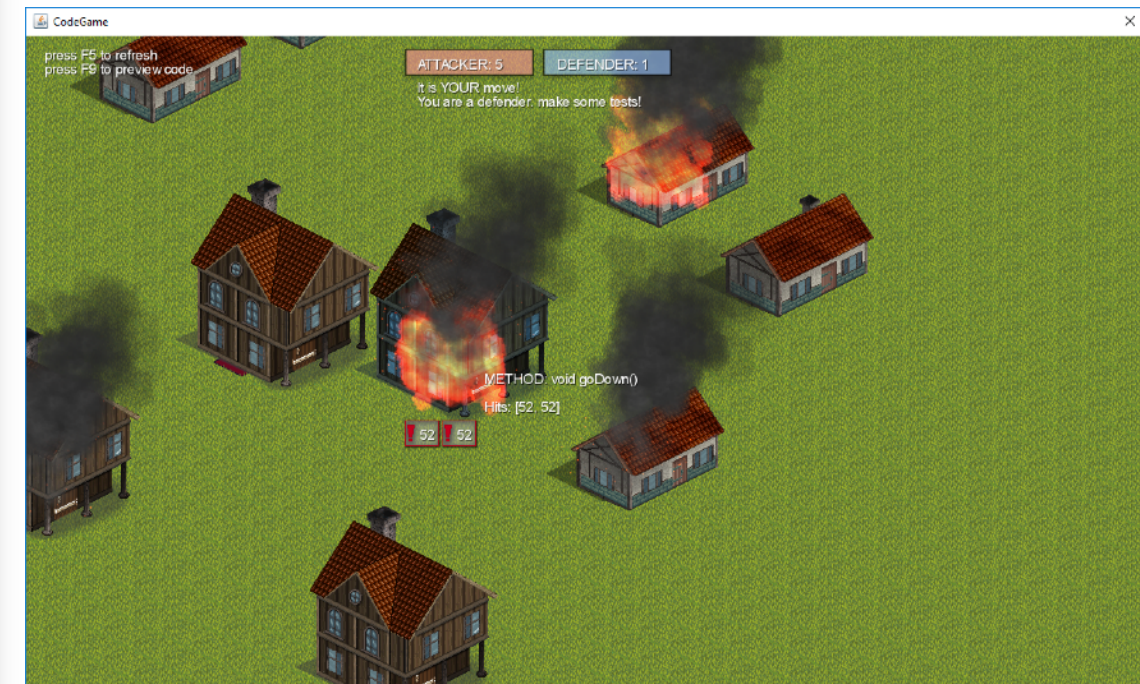
Existing Mutants

| alive (35)                                  | killed (58) | equivalent (5)   |
|---------------------------------------------|-------------|------------------|
| Mutant 5838   Creator: amin [UID: 428]      |             | Claim Equivalent |
| Mutant 5855   Creator: abrahamc2 [UID: 432] |             | Claim Equivalent |
| Mutant 5823   Creator: gregoryg [UID: 426]  |             | Claim Equivalent |
| Mutant 5886   Creator: gregoryg [UID: 426]  |             | Claim Equivalent |

JUnit tests

```
1 /** no package name */
2
3 import org.junit.*;
4 import static org.junit.Assert.*;
5
6 public class TestSparseIntArray {
7     @Test(timeout = 4000)
8     public void test() throws Throwable {
9         SparseIntArray sia = new SparseIntArray(0);
10         sia.put(0, 7);
11         sia.put(1, 8);
12         sia.delete(0);
13         assertEquals(1, sia.size());
14         assertEquals(8, sia.get(1));
15     }
16 }

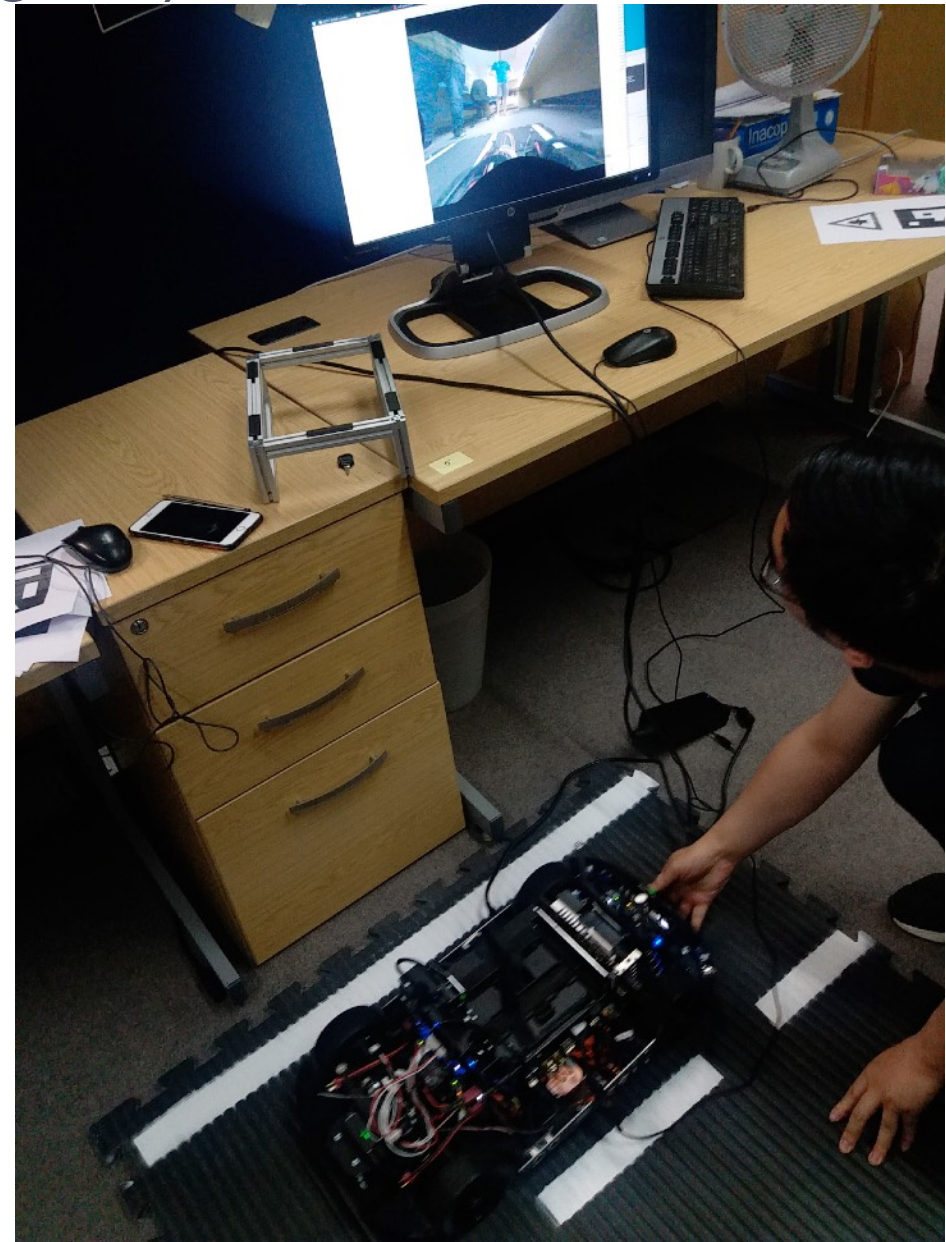
```





# Trabajo Relacionado

Pruebas automáticas de sistemas autónomos  
(e.g. self-driving car)

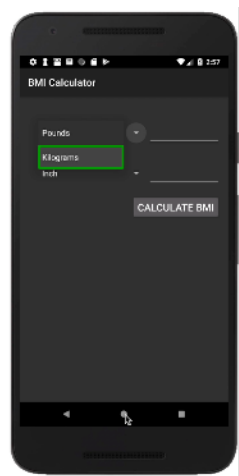


# Trabajo Relacionado

Pruebas no funcionales para apps móviles, e.g., Accesibilidad

# Trabajo Relacionado

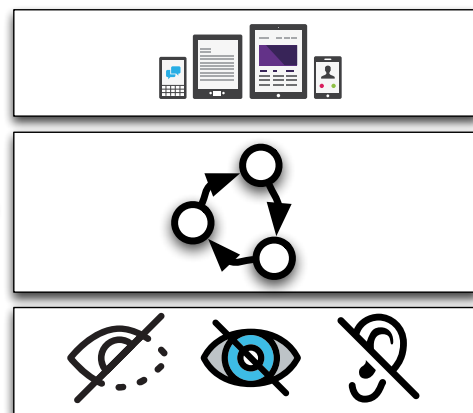
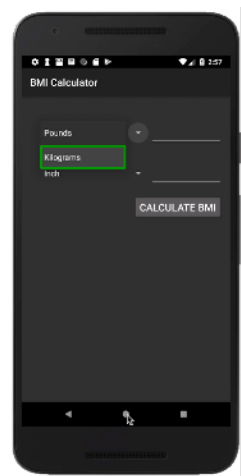
Pruebas no funcionales para apps móviles, e.g., Accesibilidad



App under test

# Trabajo Relacionado

Pruebas no funcionales para apps móviles, e.g., Accesibilidad

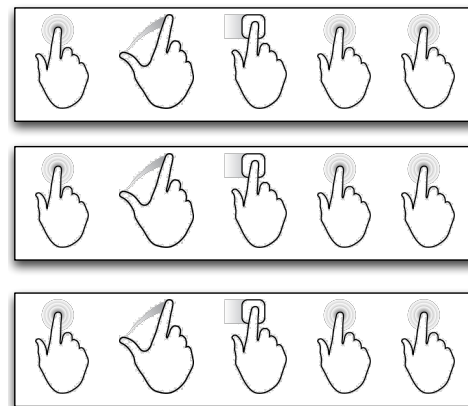
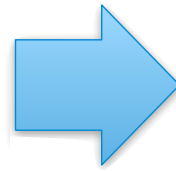
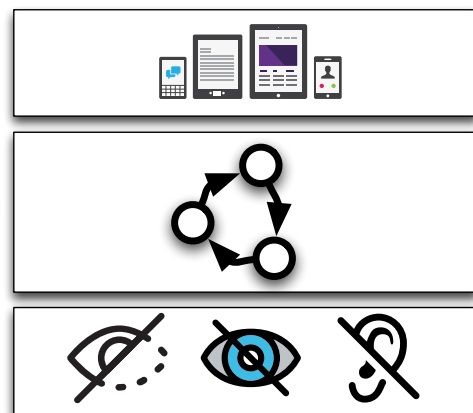
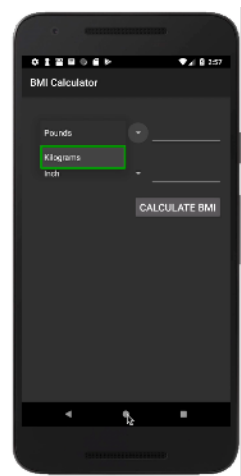


App under test

Test Generator

# Trabajo Relacionado

Pruebas no funcionales para apps móviles, e.g., Accesibilidad



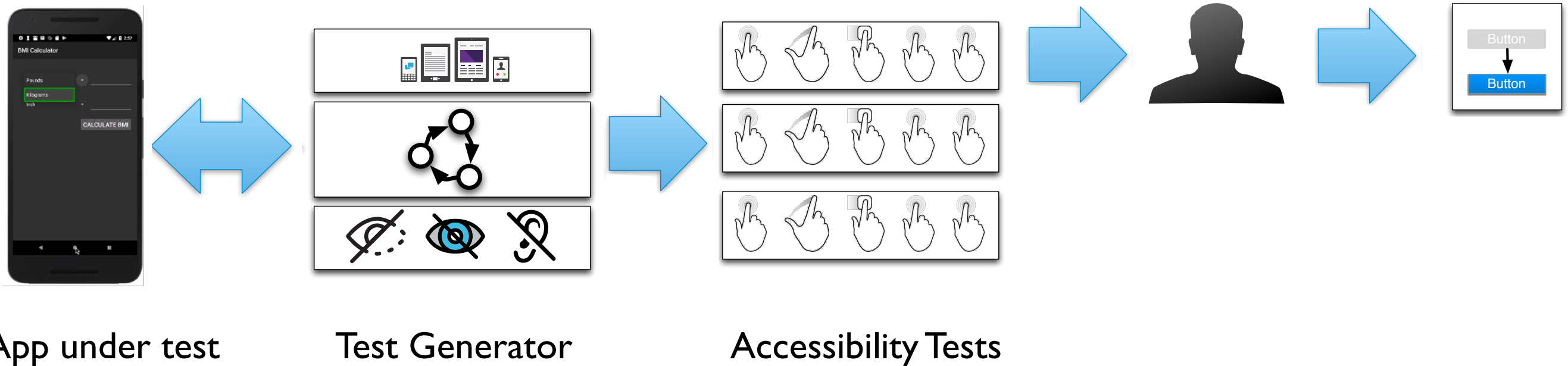
App under test

Test Generator

Accessibility Tests

# Trabajo Relacionado

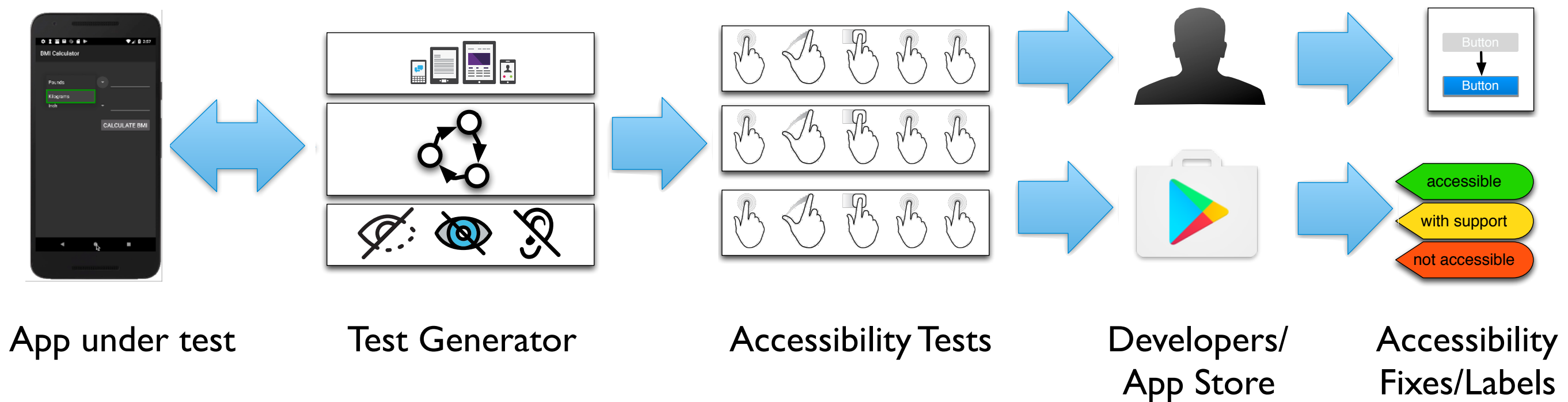
Pruebas no funcionales para apps móviles, e.g., Accesibilidad



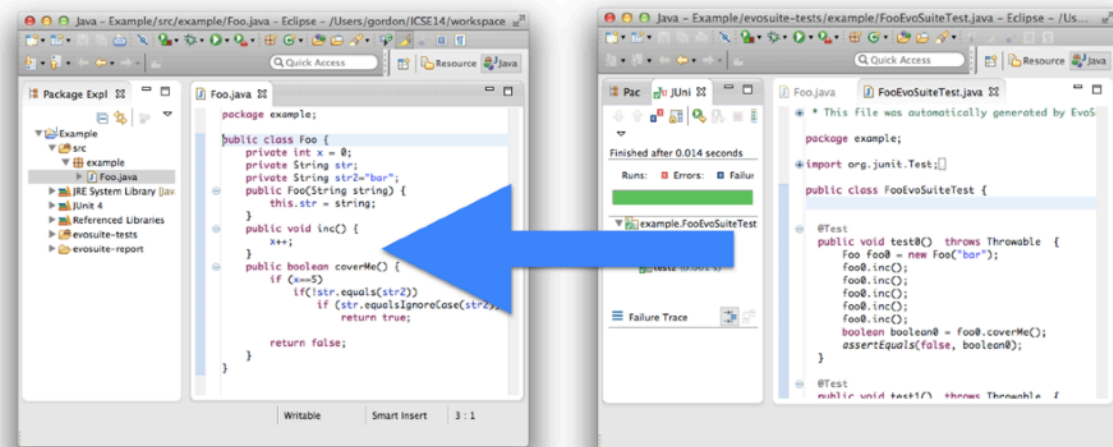


# Trabajo Relacionado

Pruebas no funcionales para apps móviles, e.g., Accesibilidad



# EvoSuite



Código Fuente

Casos de Prueba

Generación Automática



# Evaluación Empírica



Estudio en línea



47 participantes



10  
clases Java



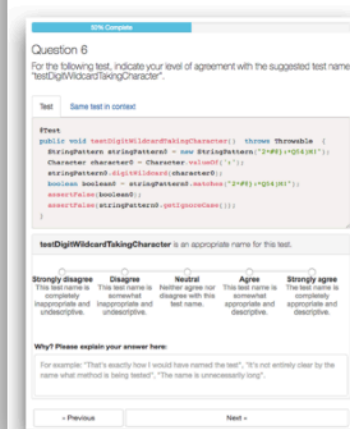
2 expertos  
como control



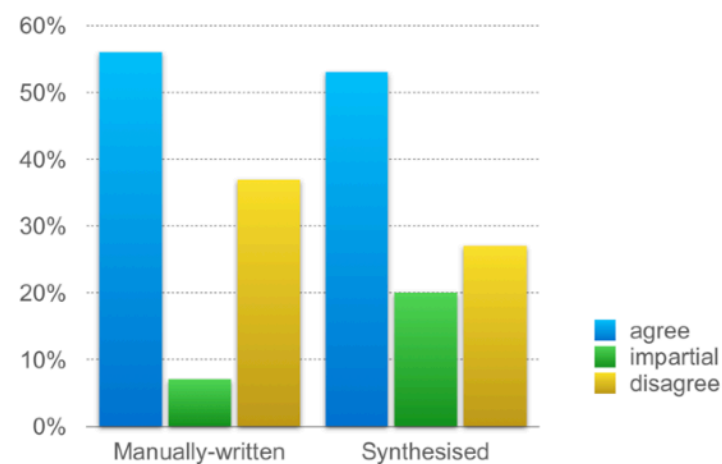
Sin tiempo límite



# Nombres de Tests



"Do you agree with the name of the following test?"



[evosuite.org](http://evosuite.org)

j.rojas@leicester.ac.uk