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

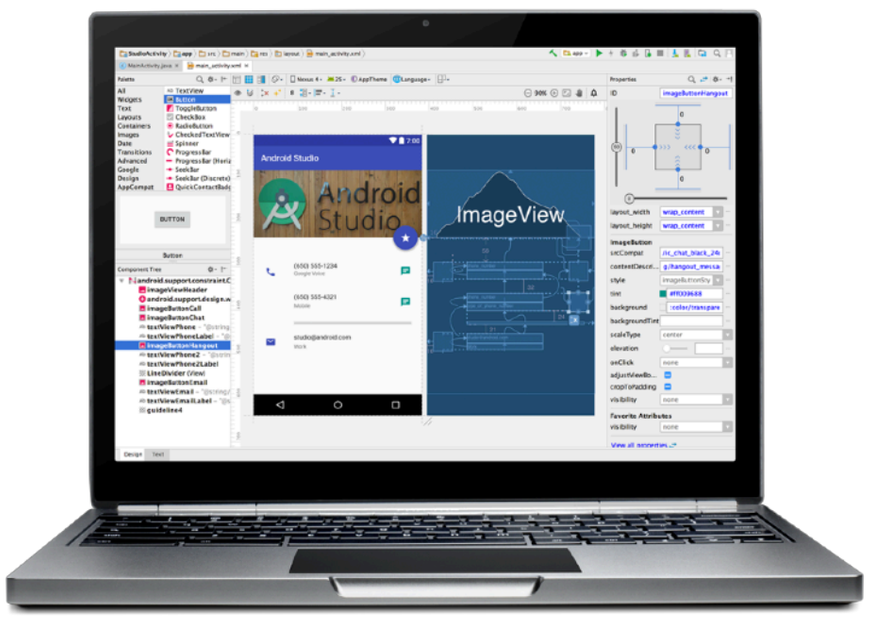
CV

- 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

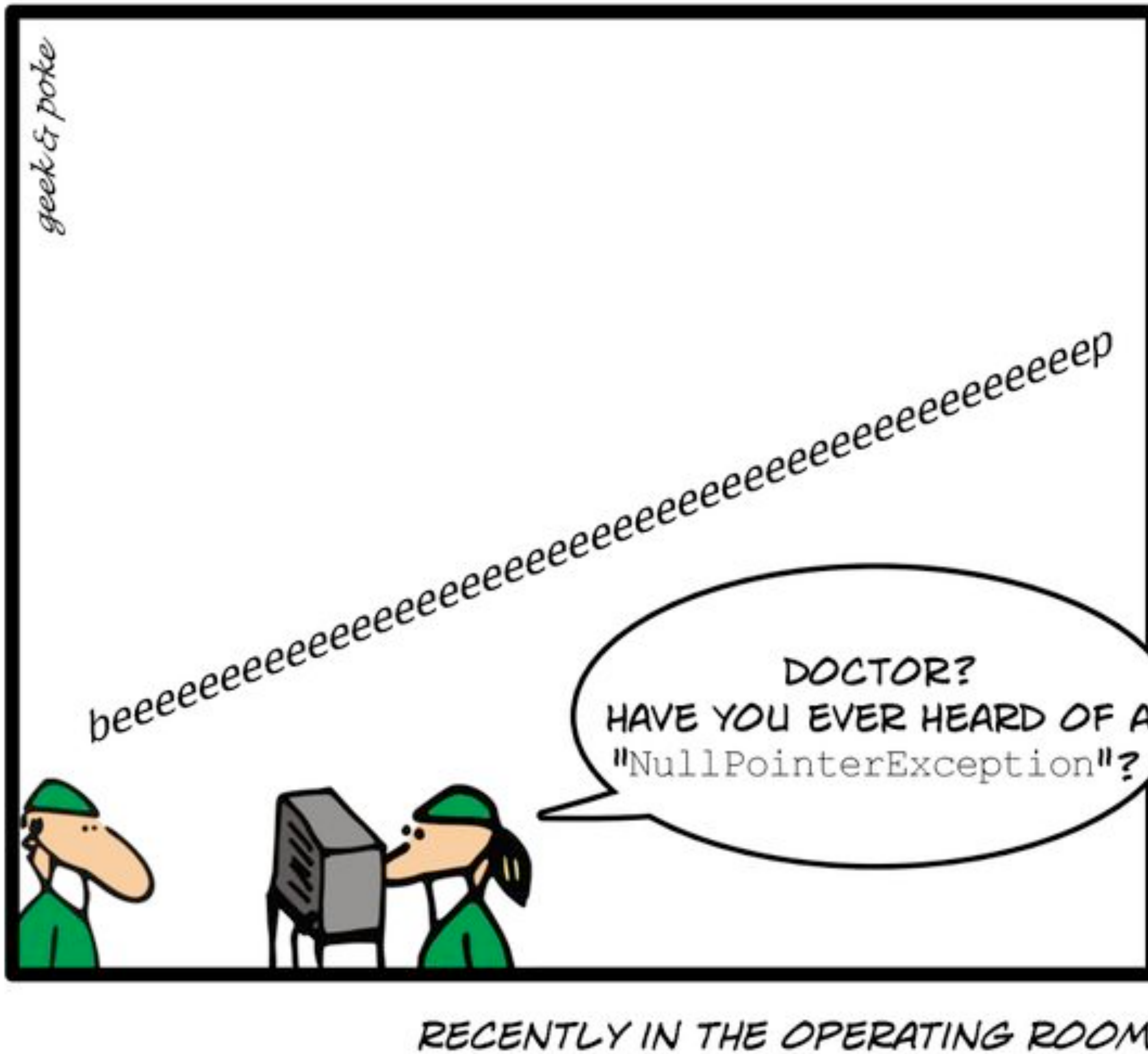
UNIVERSITY OF LEICESTER

UNIVERSITY OF
LEICESTER



UNIVERSITY OF LEICESTER

Bugs


geek & poke
beeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeep
DOCTOR?
HAVE YOU EVER HEARD OF A "NullPointerException"?
RECENTLY IN THE OPERATING ROOM

Windows

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

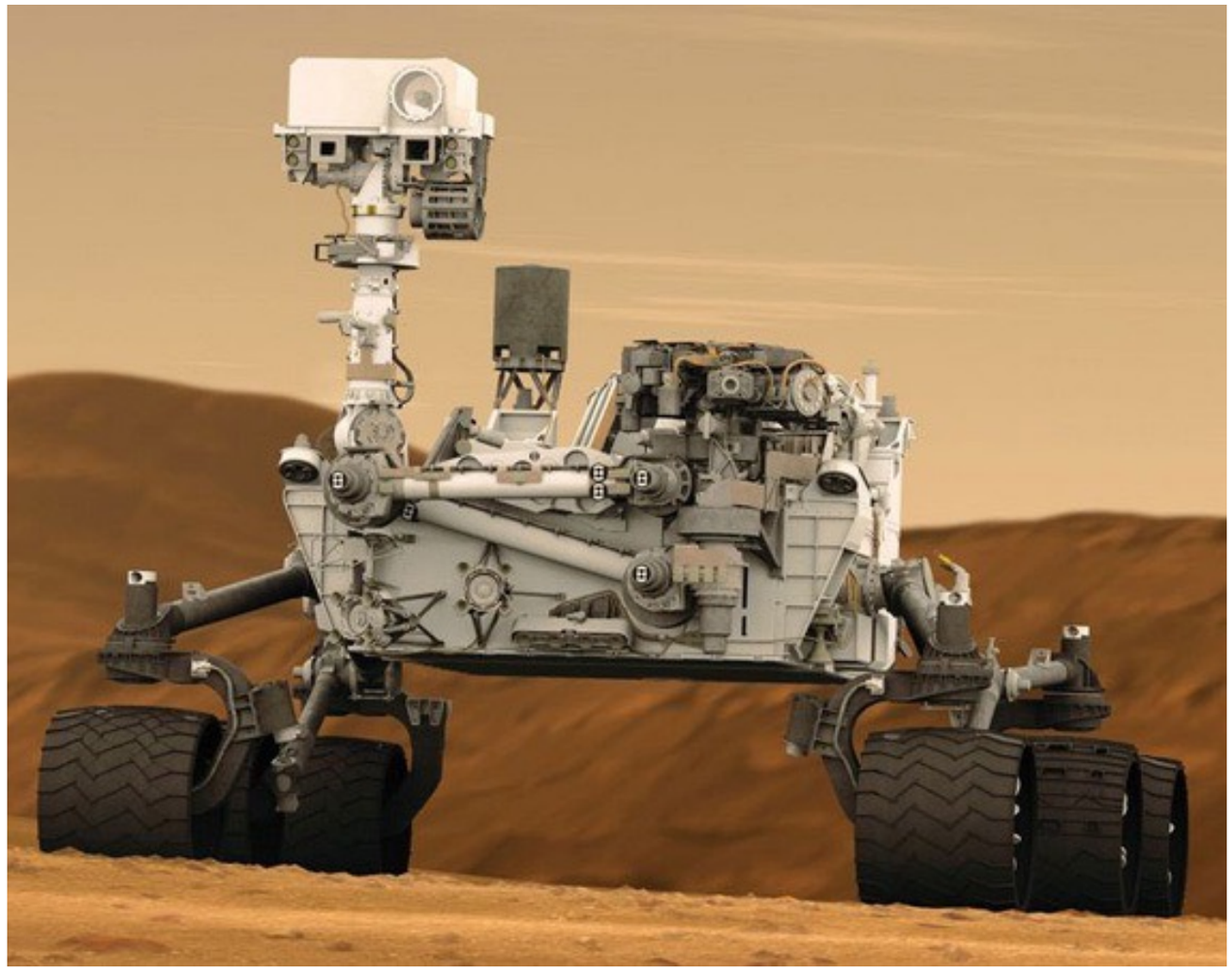
* Press any key to terminate the application.
* Press CTRL+ALT+DEL to restart your computer. You will
  lose any unsaved information in all applications.

Press any key to continue

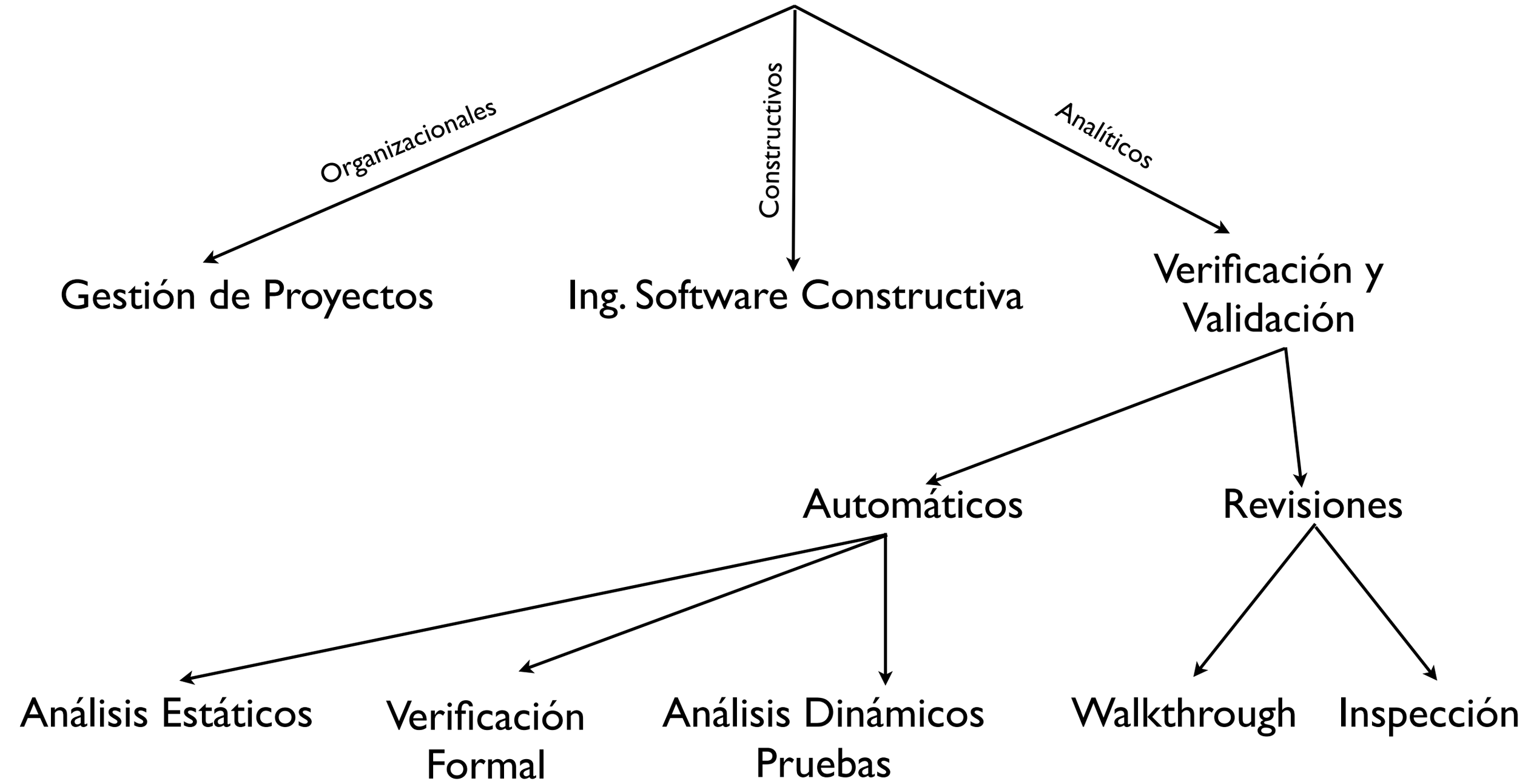
Ariane 5, primer vuelo, 1996

Therac-25, 1985-87

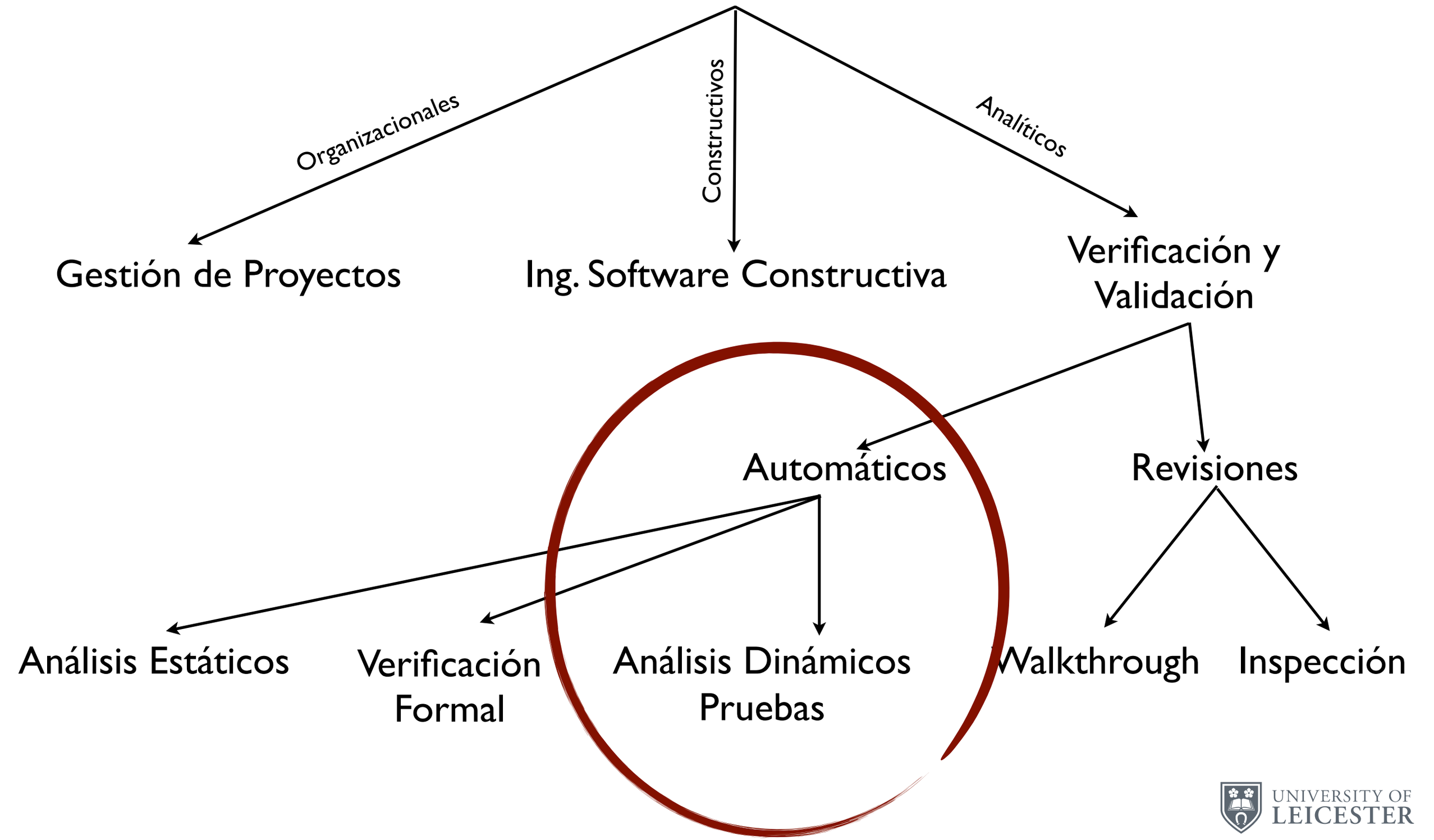
¿Software libre de errores?



Métodos para la Calidad del Software

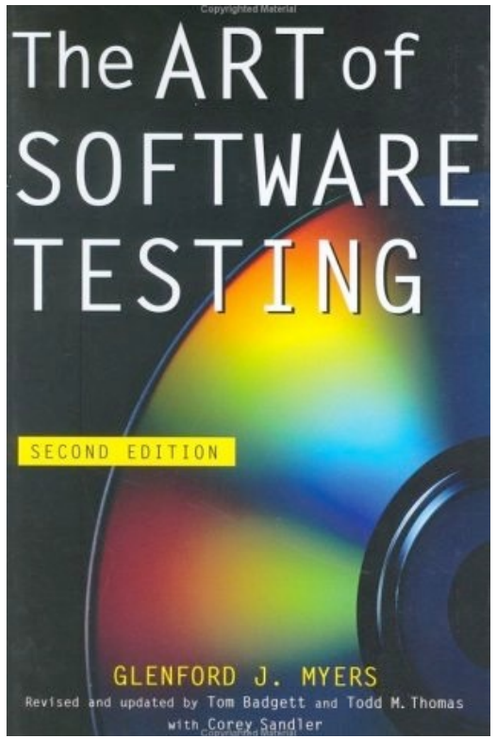


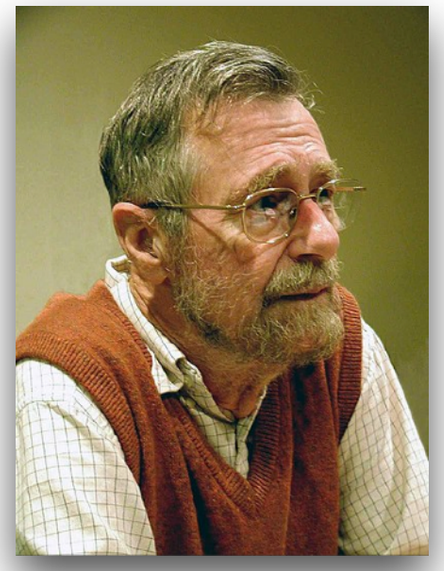
Métodos para la Calidad del Software



Pruebas de Software

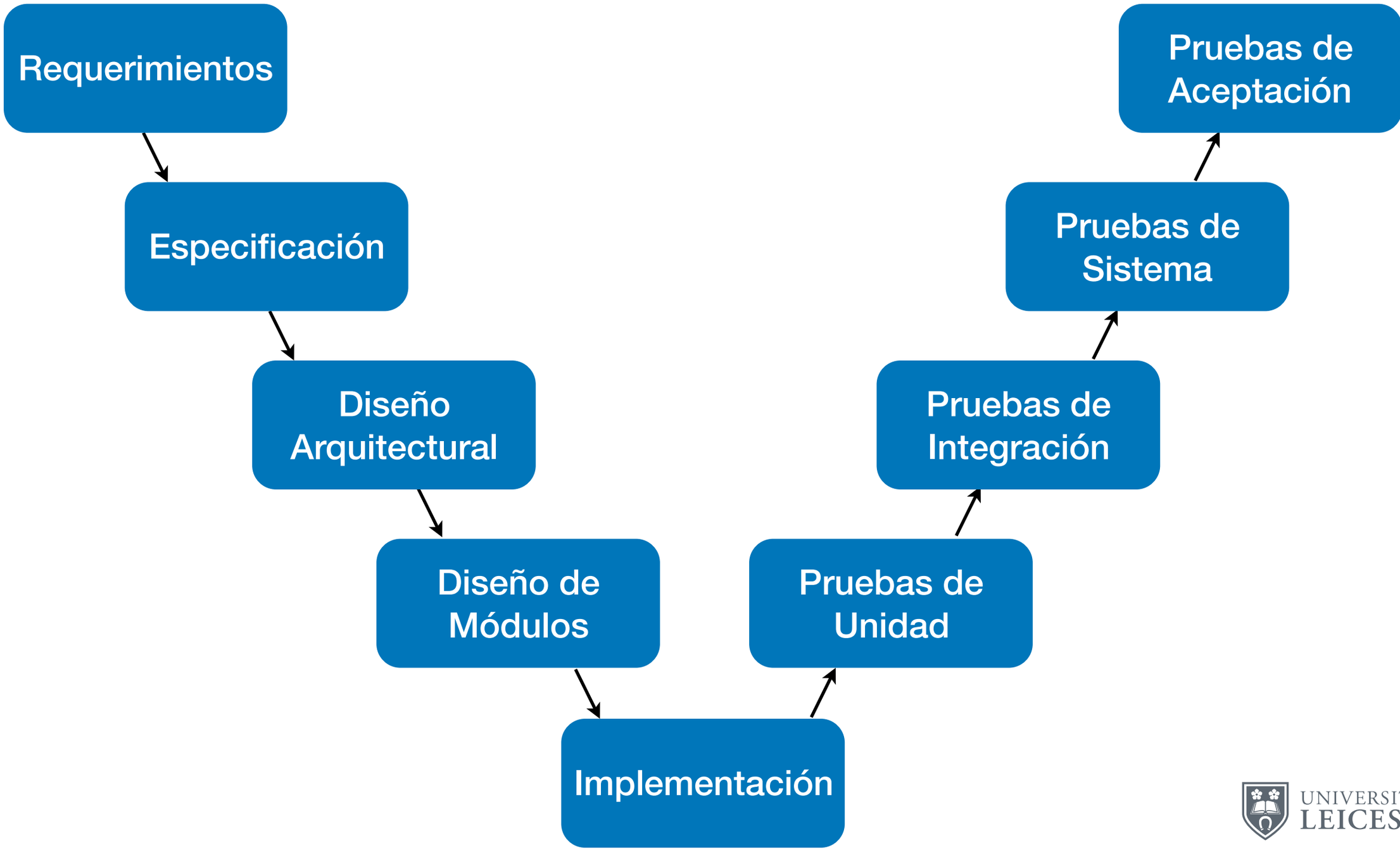
“La selección de entradas para un sistema con la intención de provocar fallos en dicho sistema, revelando de tal forma la presencia de errores”



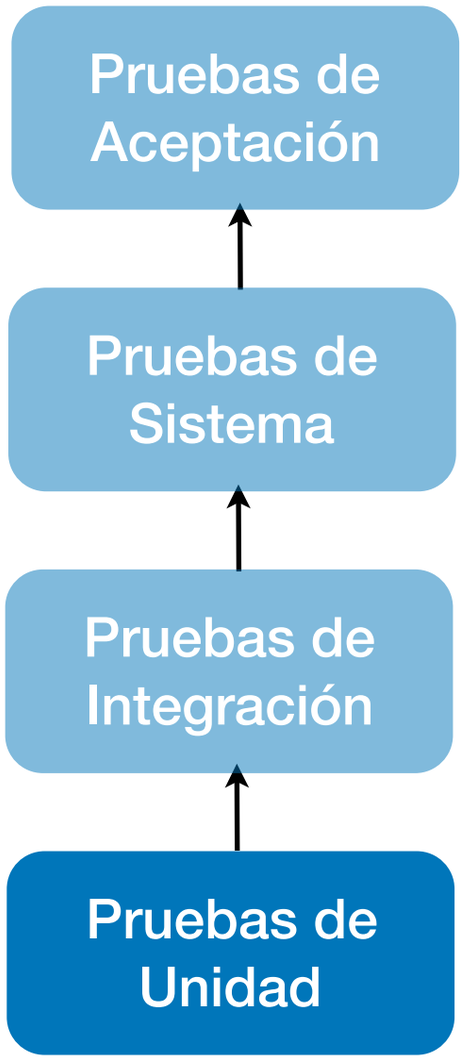


”Las pruebas de software solo pueden demostrar la presencia de errores, no su ausencia.” (E. Dijkstra)

Pruebas de Software

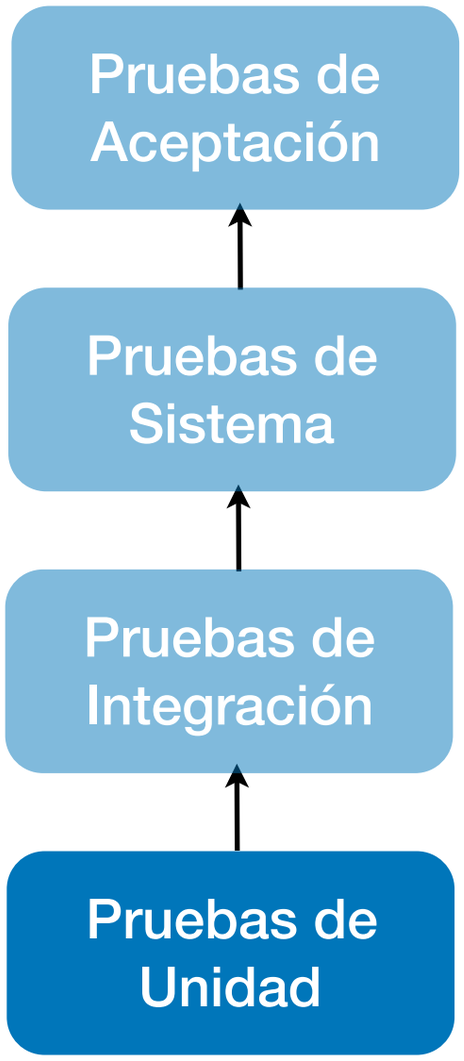


Pruebas de Unidad

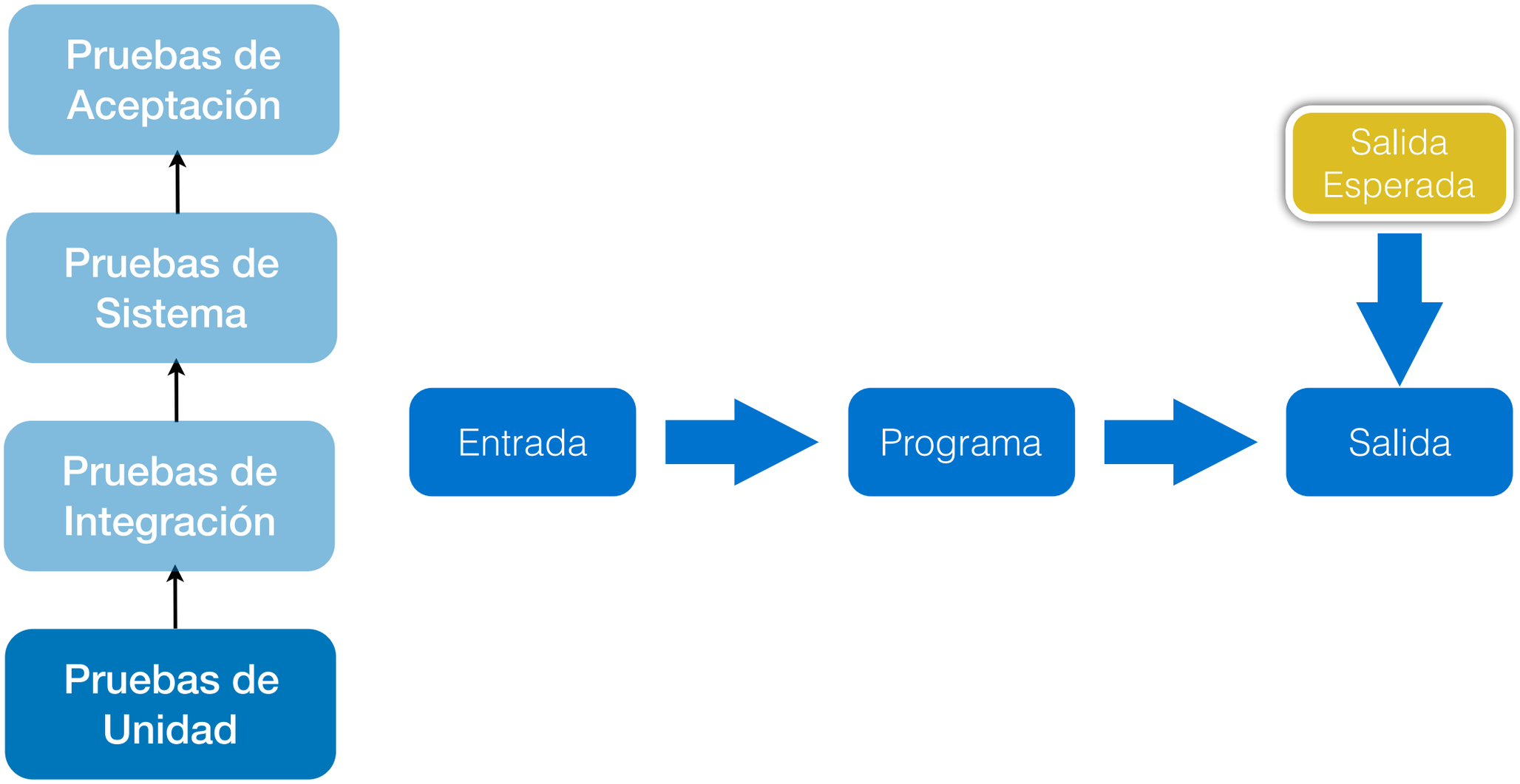


1. Ejecutar programa, usando datos de prueba (*input)*
2. Verificar salida del programa (*output*), usando oráculo de pruebas (*test oracle*)

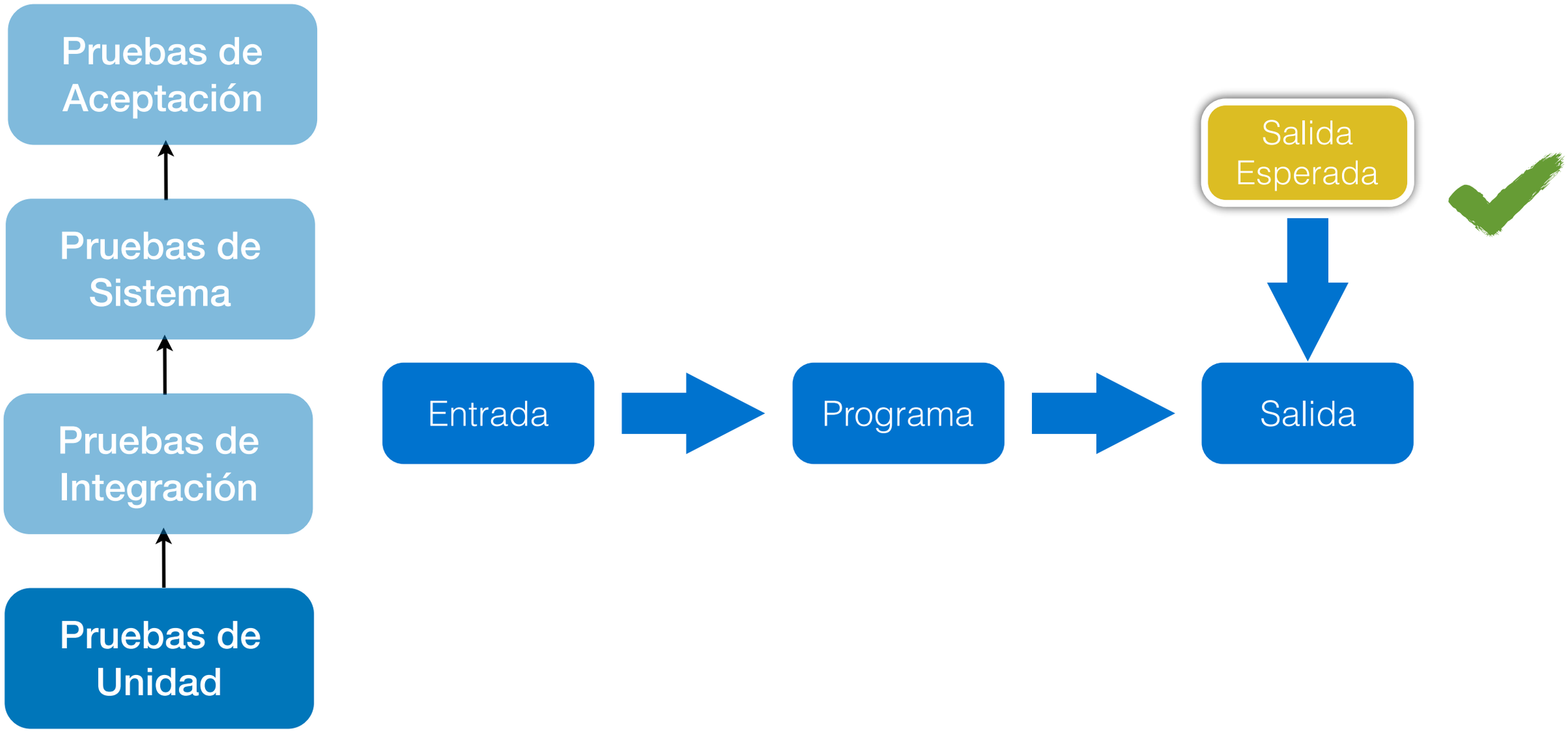
Pruebas de Unidad



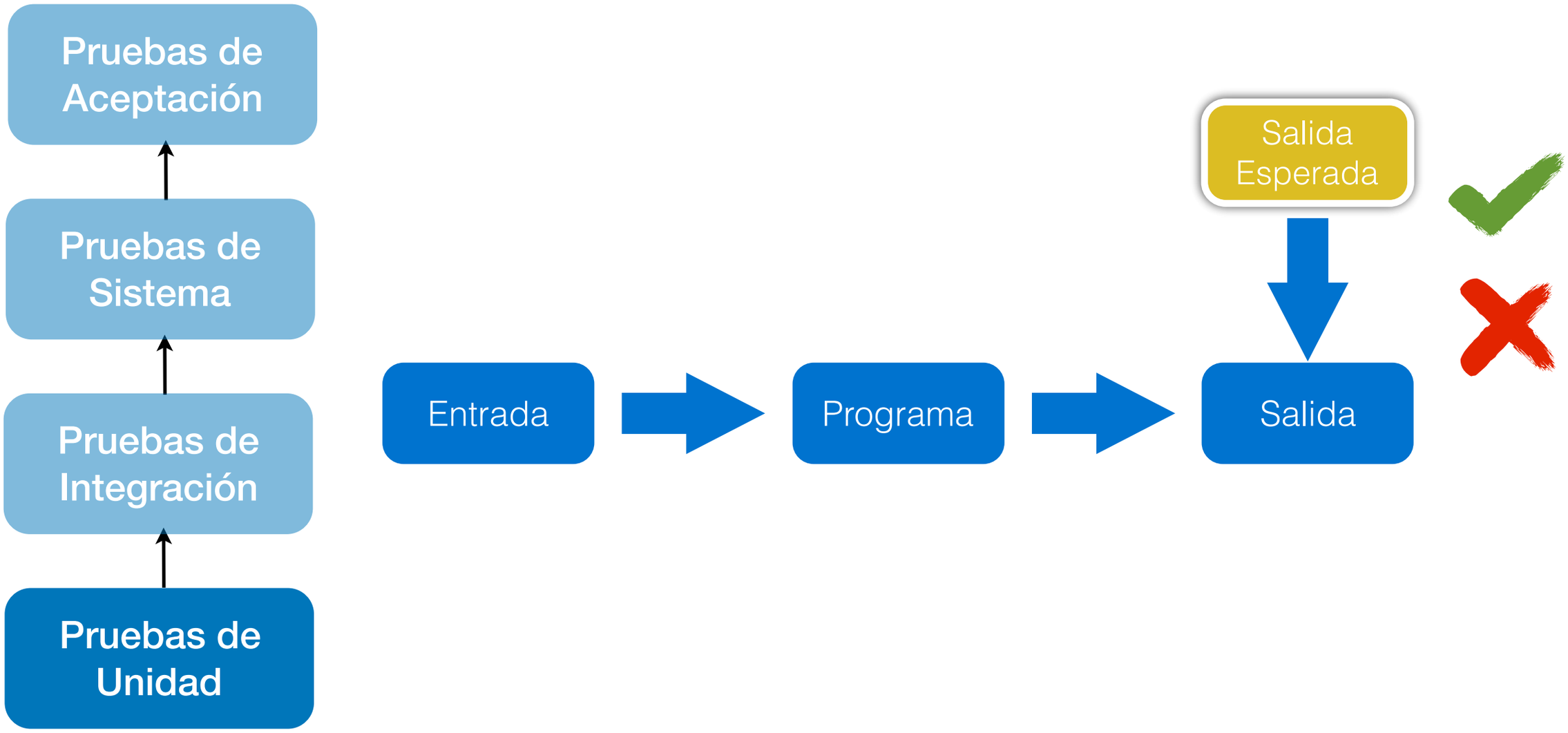
Pruebas de Unidad



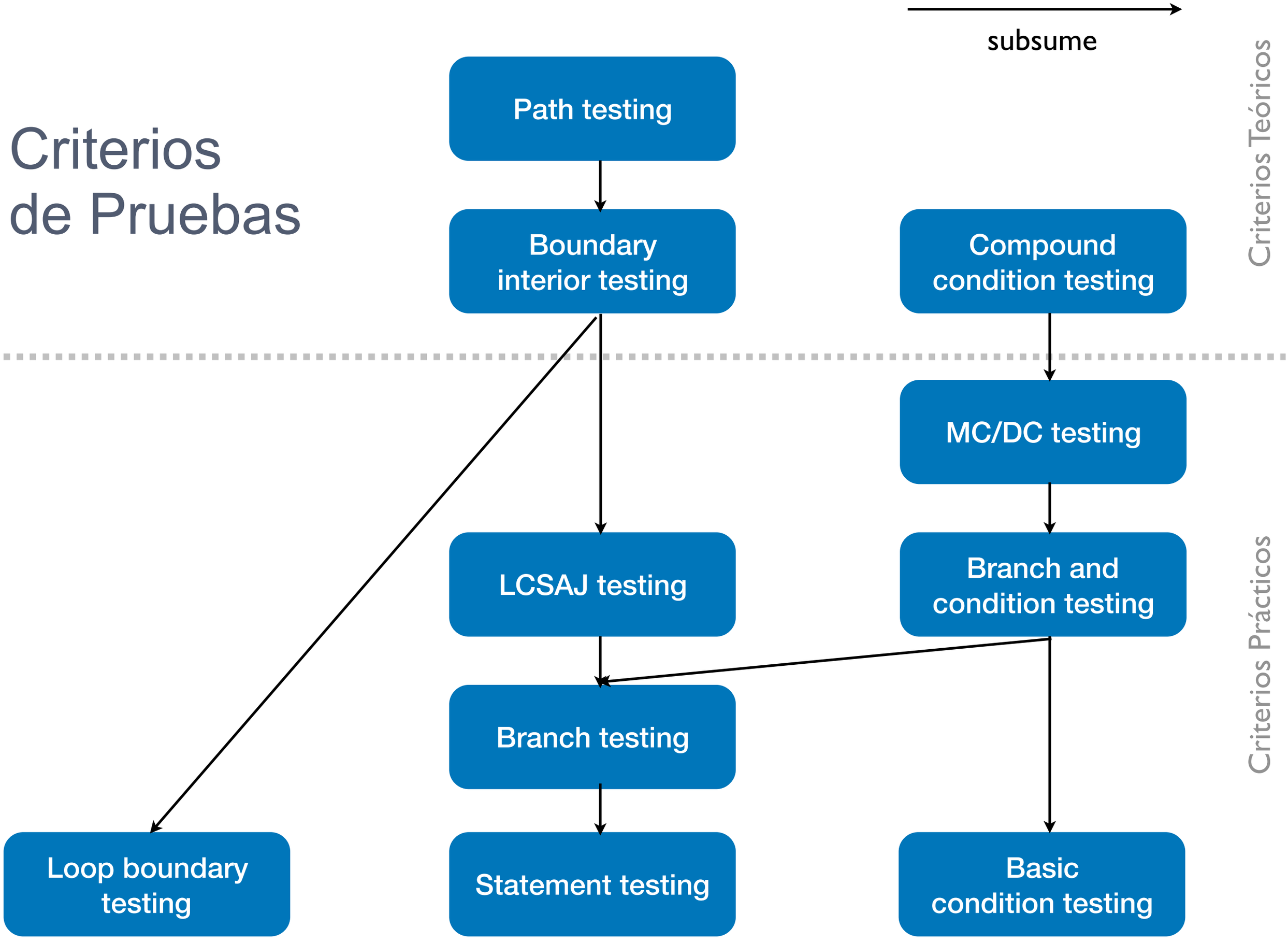
Pruebas de Unidad



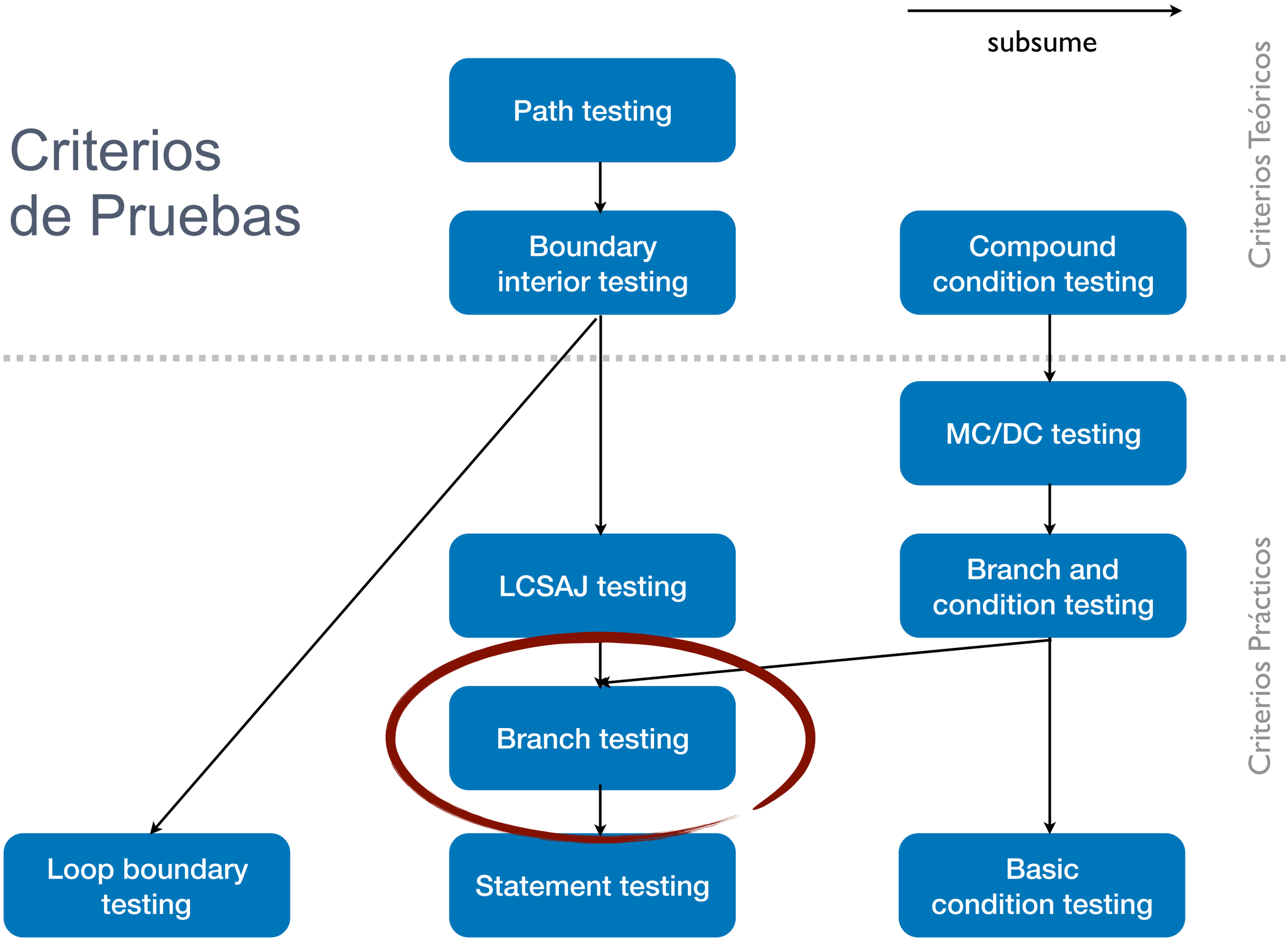
Pruebas de Unidad

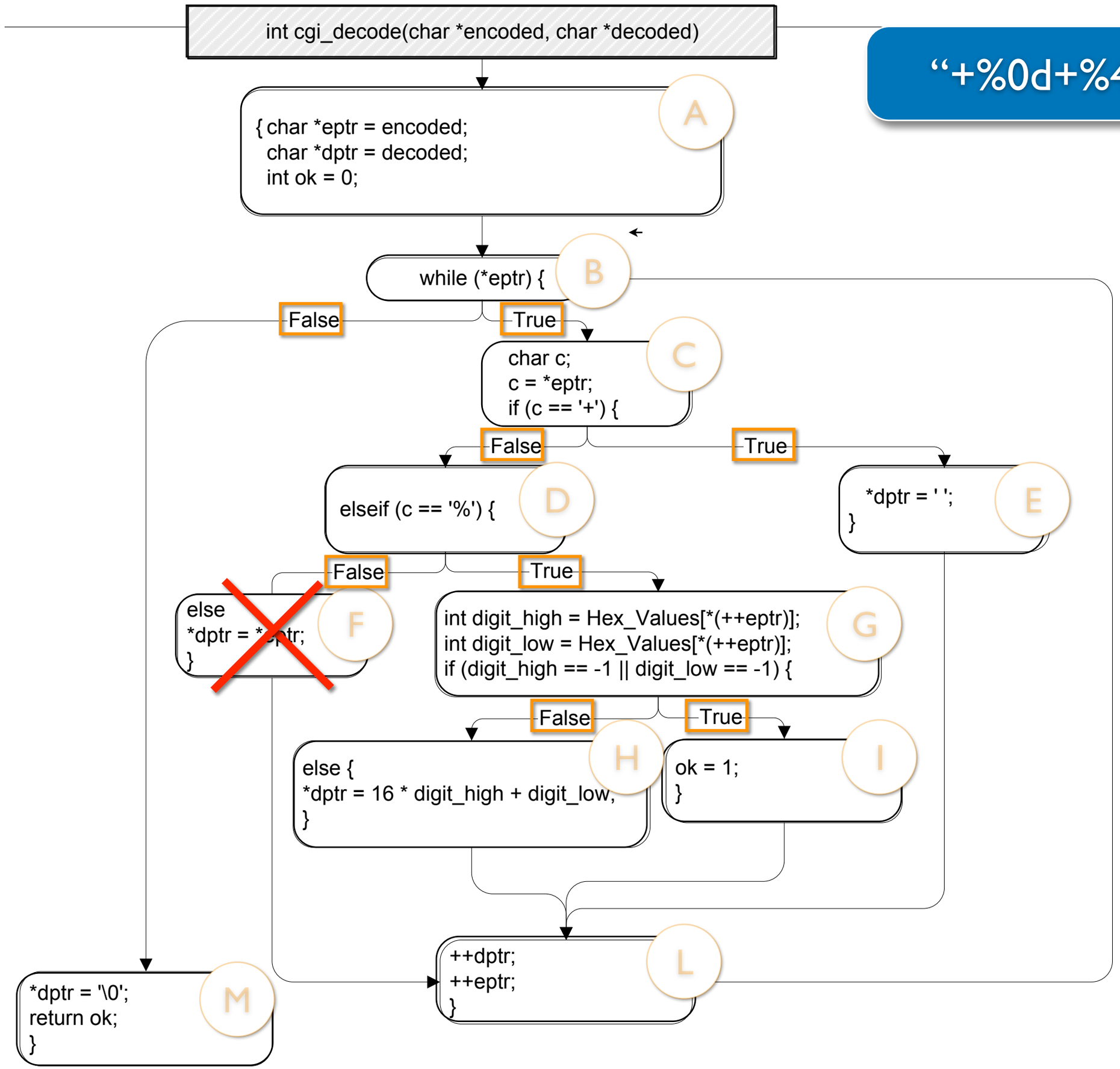


Criterios de Pruebas

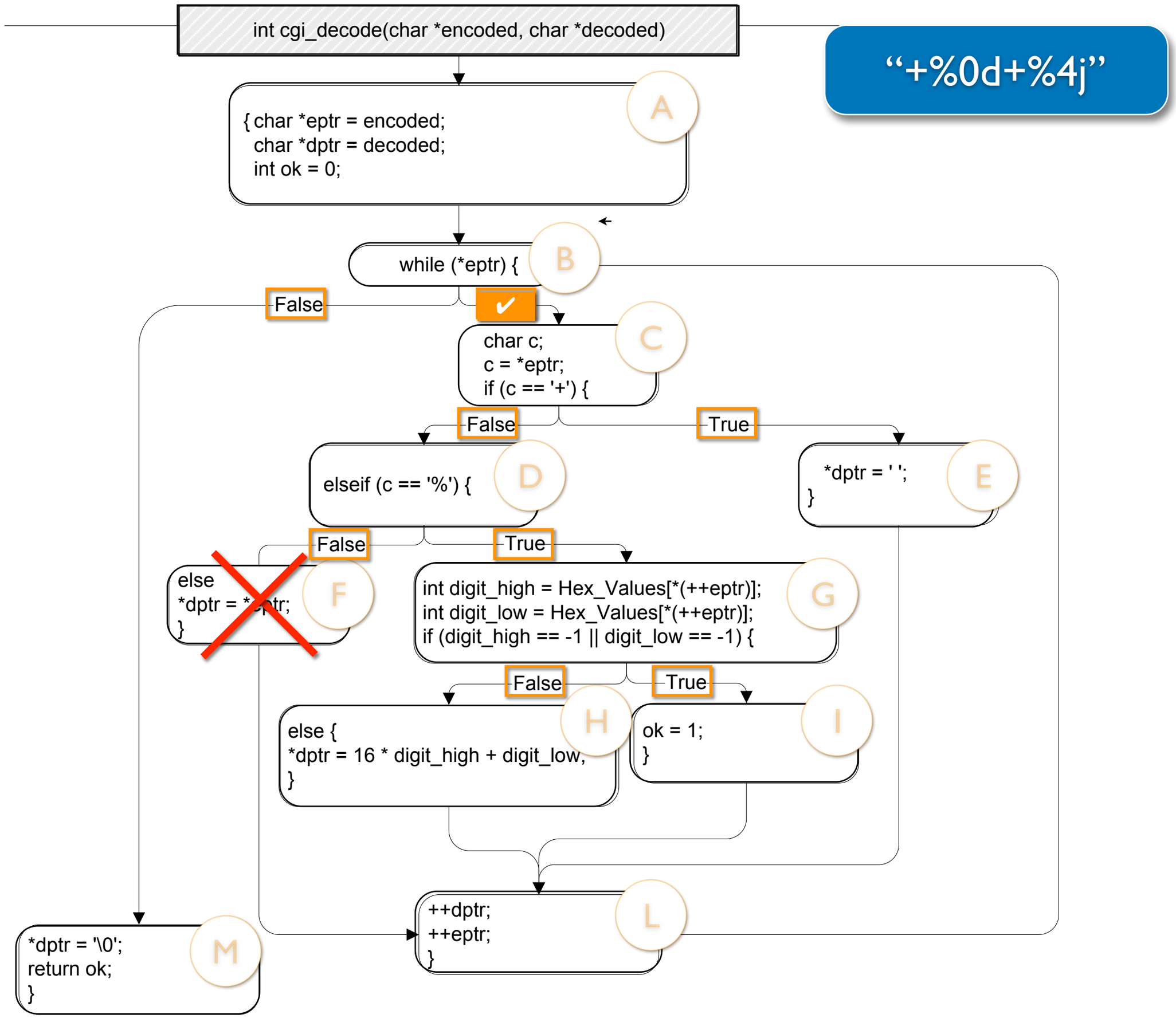


Criterios de Pruebas

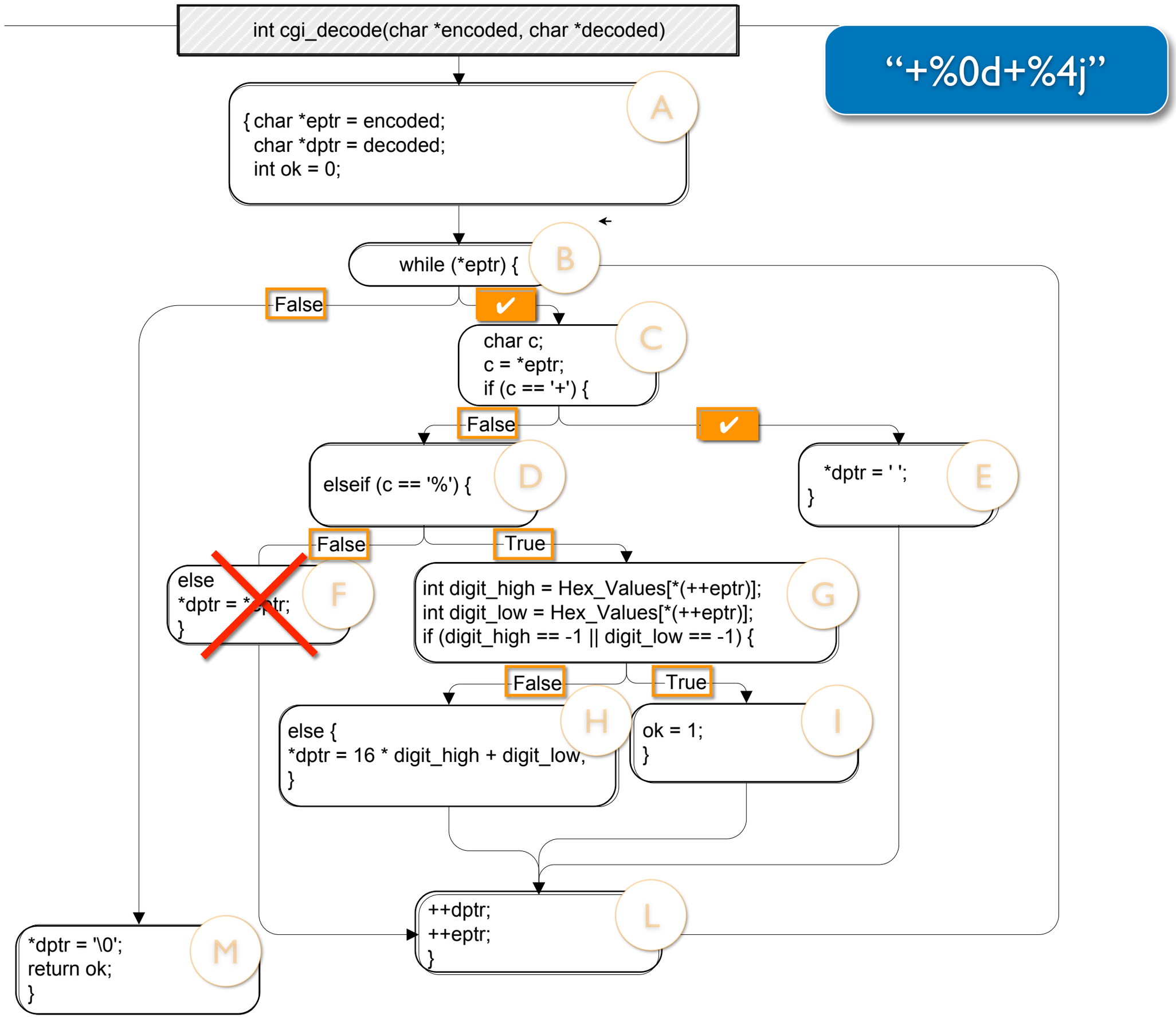


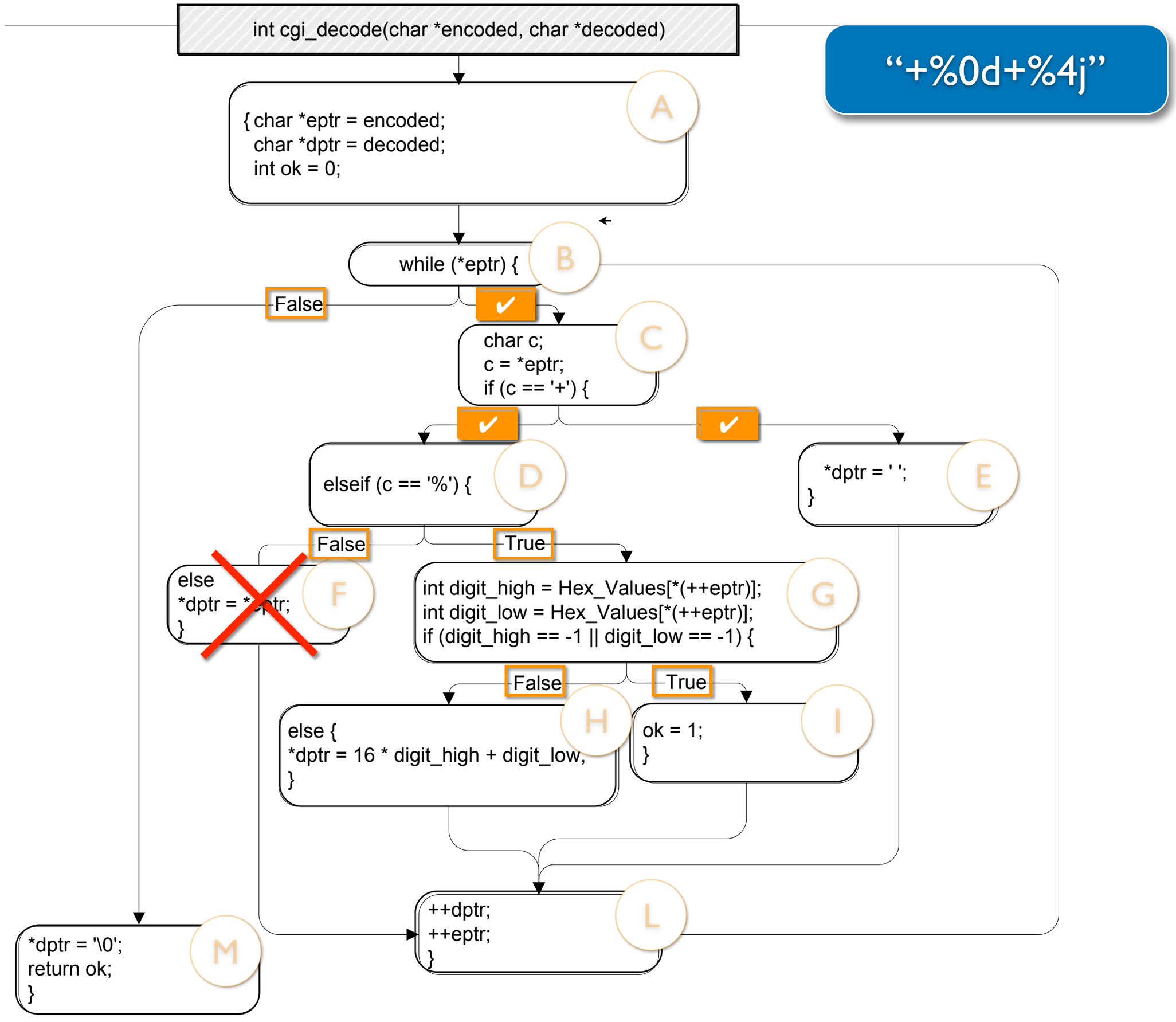

int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
False
True
char c;
c = *eptr;
if (c == '+') {
C
False
True
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
True
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
False
True
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%0d+%4j"


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
False
✔
char c;
c = *eptr;
if (c == '+') {
C
False
True
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
True
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
False
True
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%0d+%4j"


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
False
✔
char c;
c = *eptr;
if (c == '+') {
C
False
✔
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
True
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
False
True
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M
“+%0d+%4j”


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
False
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
True
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
False
True
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%0d+%4j"

“+%0d+%4j”

```
int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
  char *dptr = decoded;
  int ok = 0;
while (*eptr) {
char c;
c = *eptr;
if (c == '+') {
*dptr = ' ';
}
elseif (c == '%') {
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
else {
*dptr = 16 * digit_high + digit_low;
}
ok = 1;
}
else
*dptr = *eptr;
}
++dptr;
++eptr;
}
*dptr = '\0';
return ok;
}
```

A B C D E F G H I L M

False ✔ ✔ ✔ False ✔ False True

“+%0d+%4j”

```
int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
  char *dptr = decoded;
  int ok = 0;
while (*eptr) {
char c;
c = *eptr;
if (c == '+') {
*dptr = ' ';
}
elseif (c == '%') {
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
else
*dptr = *eptr;
}
else {
*dptr = 16 * digit_high + digit_low;
}
ok = 1;
}
++dptr;
++eptr;
}
*dptr = '\0';
return ok;
}
```

A B C D E F G H I L M

False ✔ ✔ ✔ False ✔ ✔ True

“+%0d+%4j”

int cgi_decode(char *encoded, char *decoded)

A
```
{ char *eptr = encoded;
  char *dptr = decoded;
  int ok = 0;
```

B
```
while (*eptr) {
```

False ✔

C
```
char c;
c = *eptr;
if (c == '+') {
```

✔ ✔

D
```
elseif (c == '%') {
```

E
```
  *dptr = ' ';
}
```

False ✔

F
```
else
*dptr = *eptr;
}
```

G
```
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
```

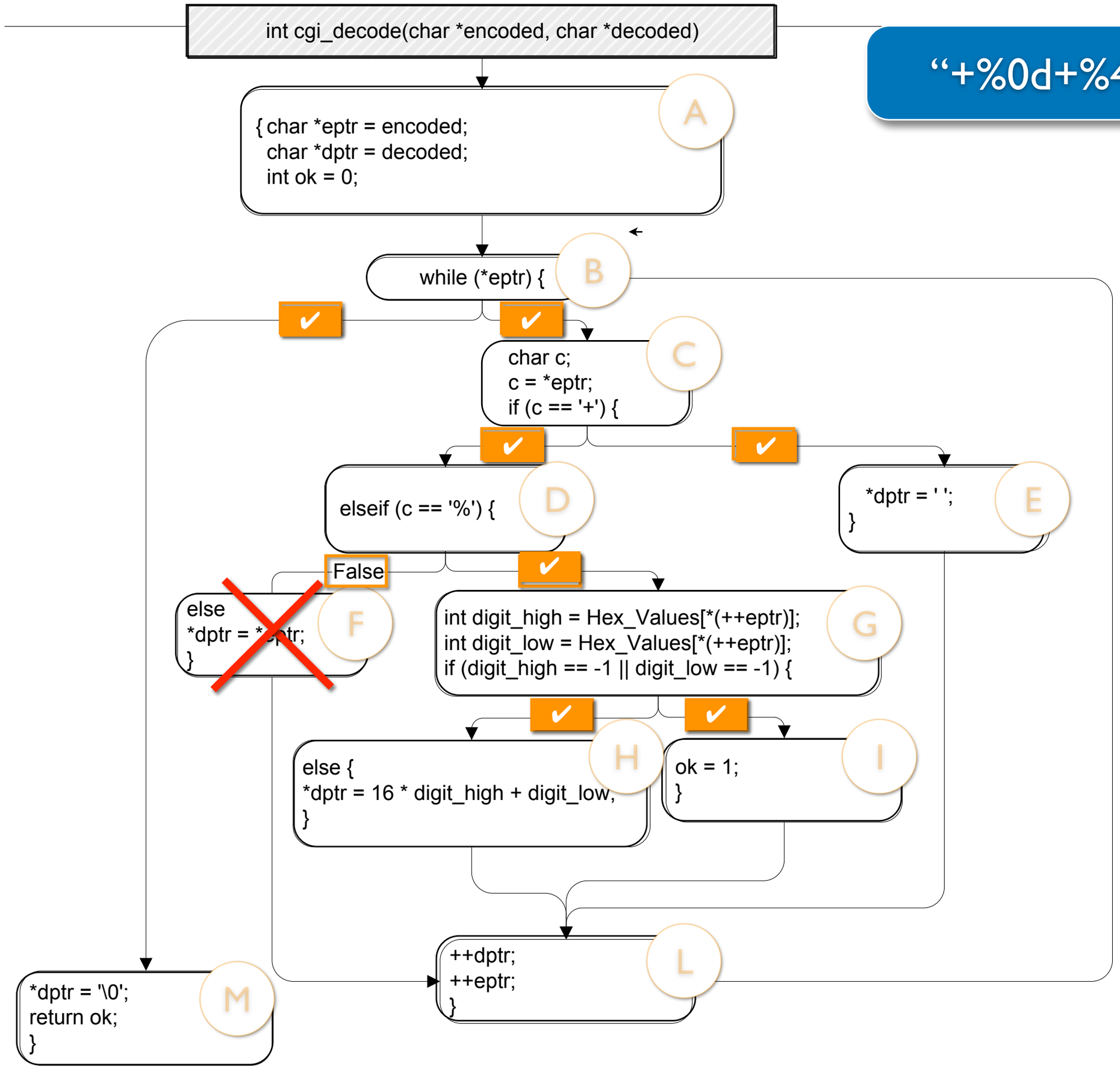
✔ ✔

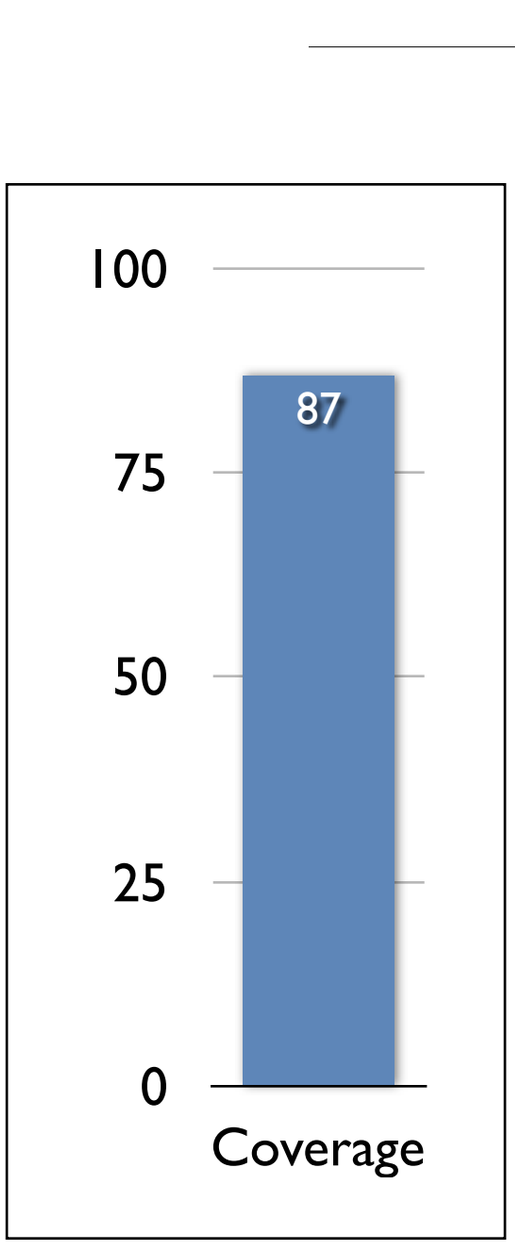
H
```
else {
*dptr = 16 * digit_high + digit_low;
}
```

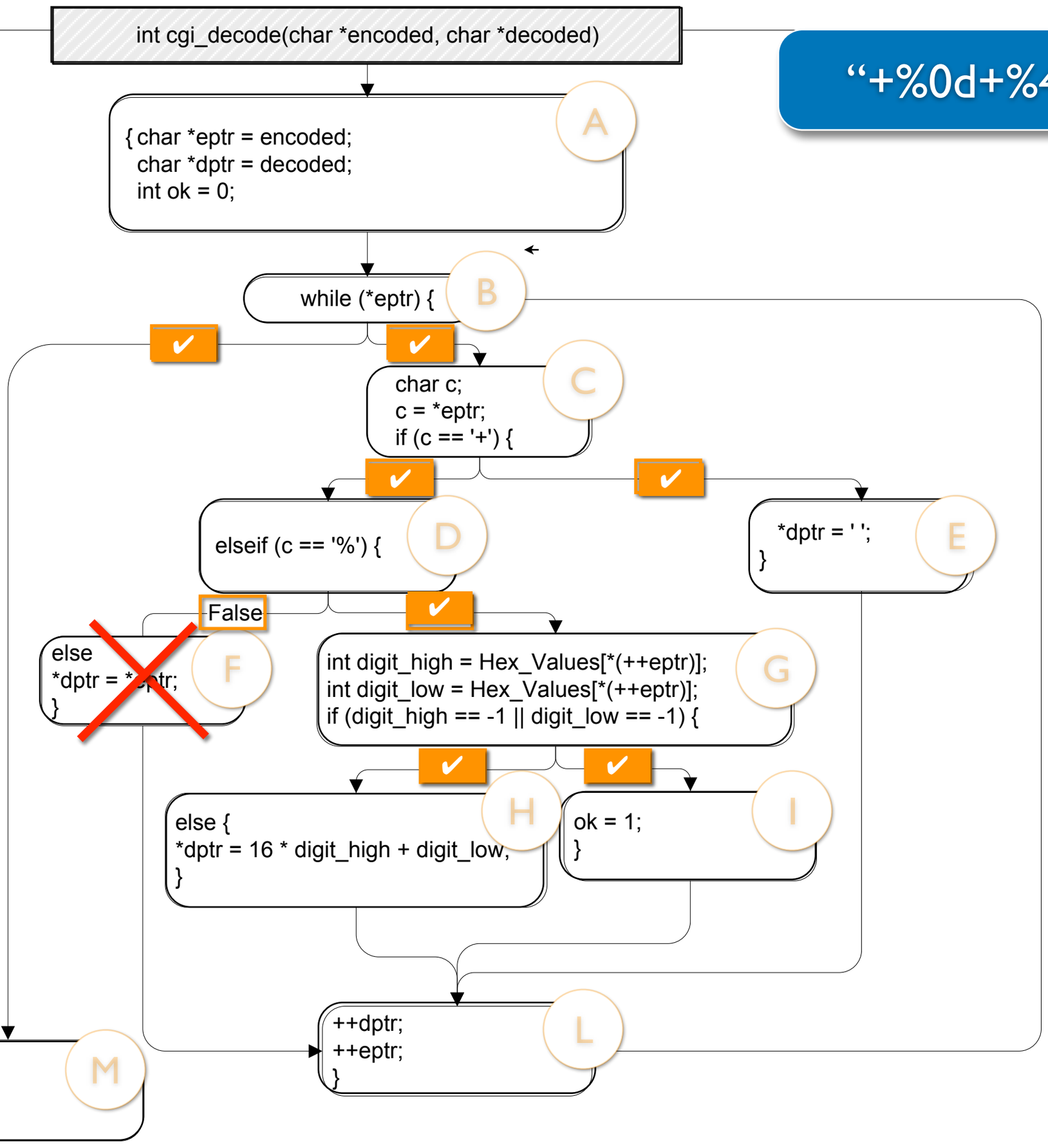
I
```
ok = 1;
}
```

L
```
++dptr;
++eptr;
}
```

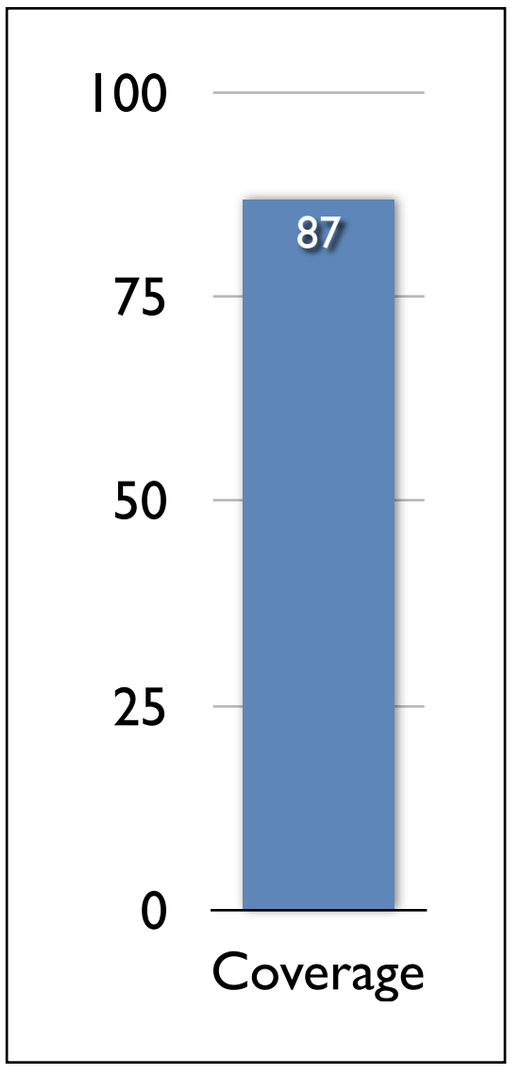
M
```
*dptr = '\0';
return ok;
}
```

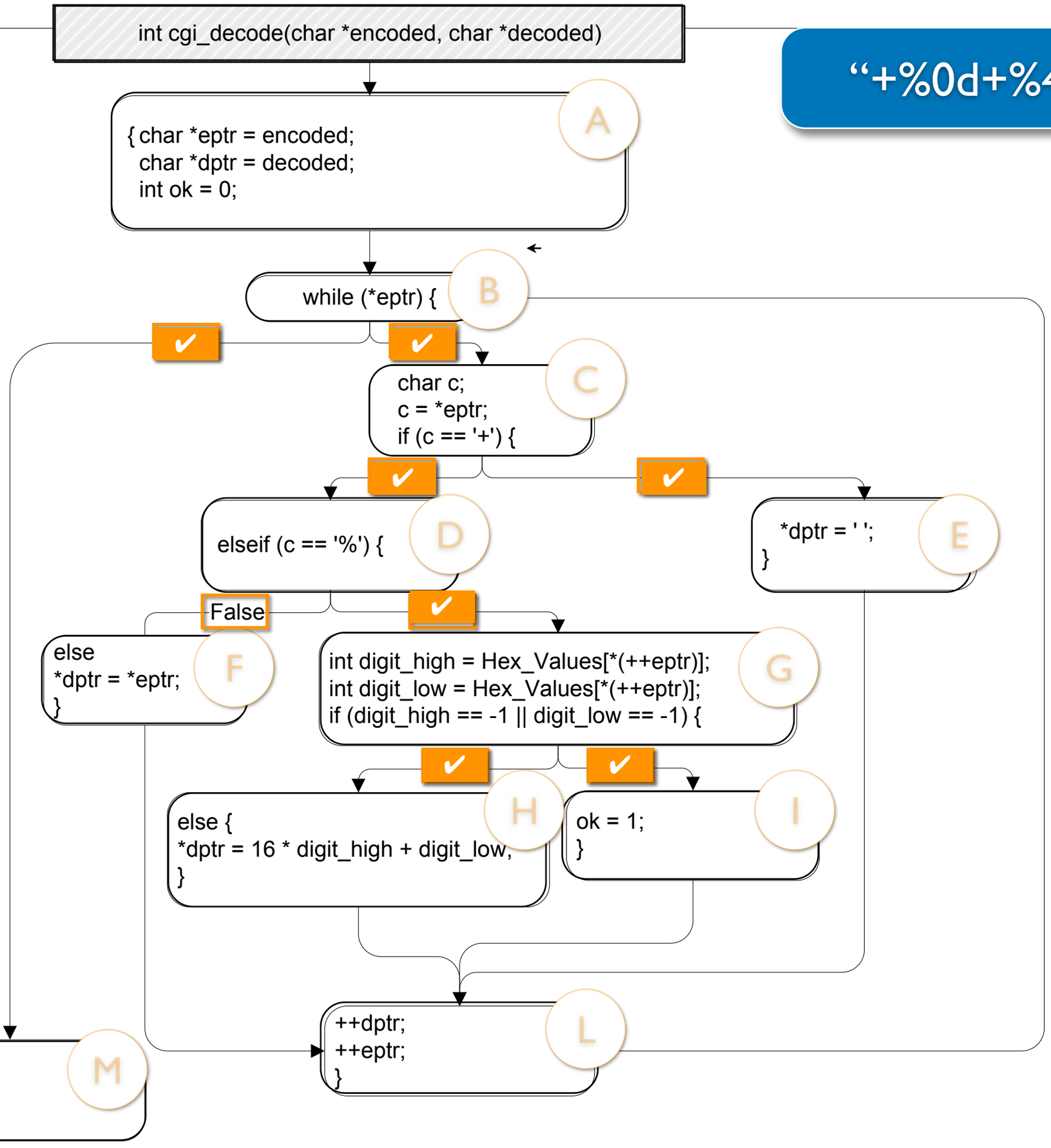

int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M
"+%0d+%4j"


100
75
50
25
0
87
Coverage

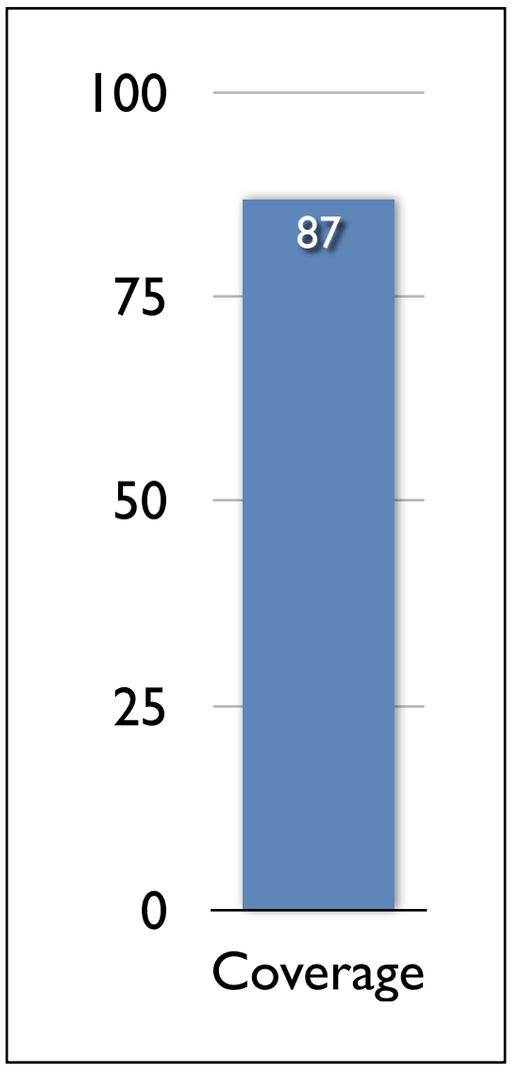

int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

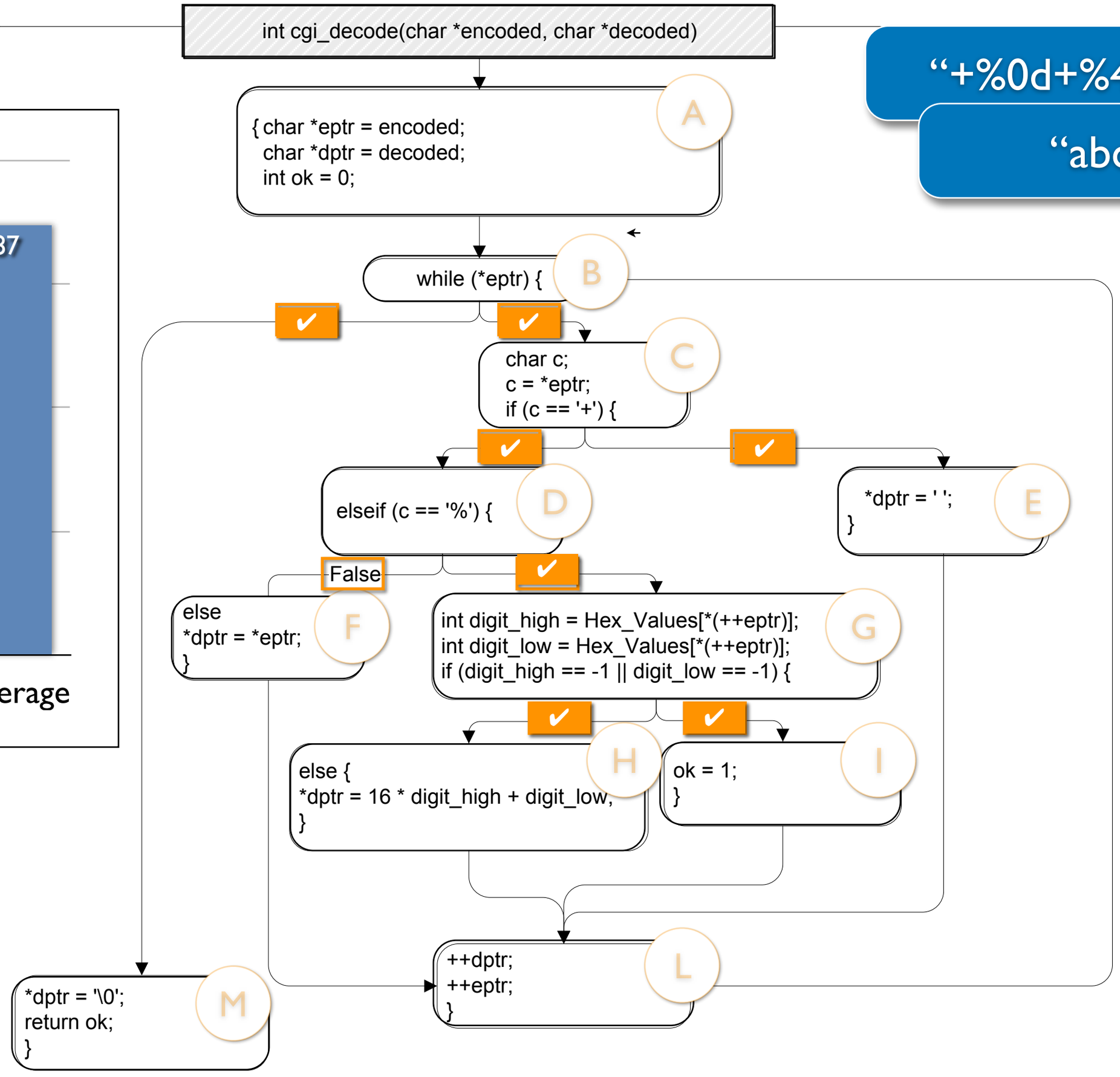
"+%0d+%4j"


100
75
50
25
0
87
Coverage


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

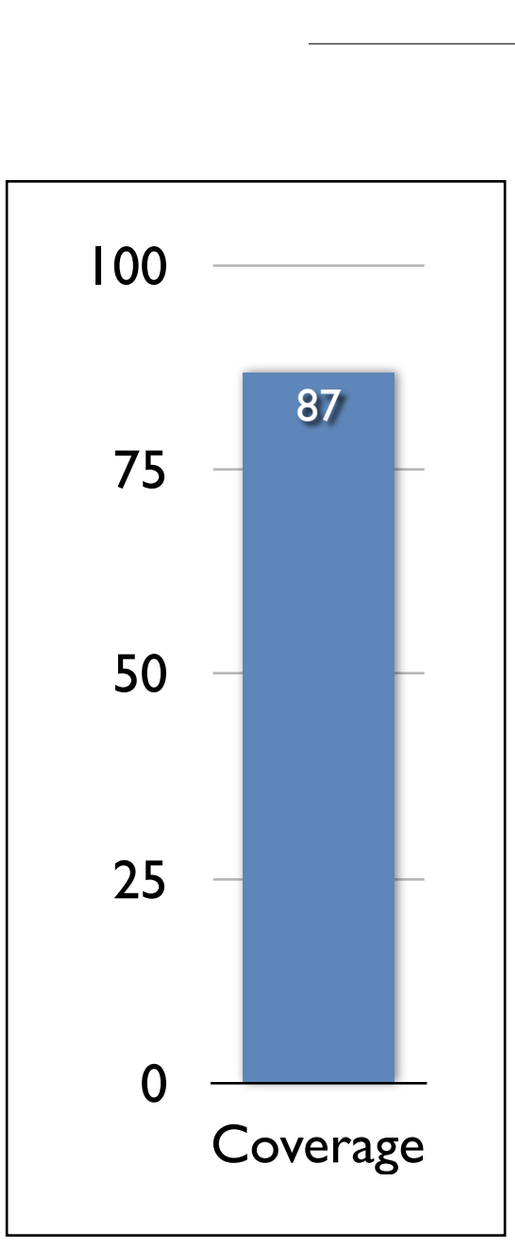
"+%0d+%4j"

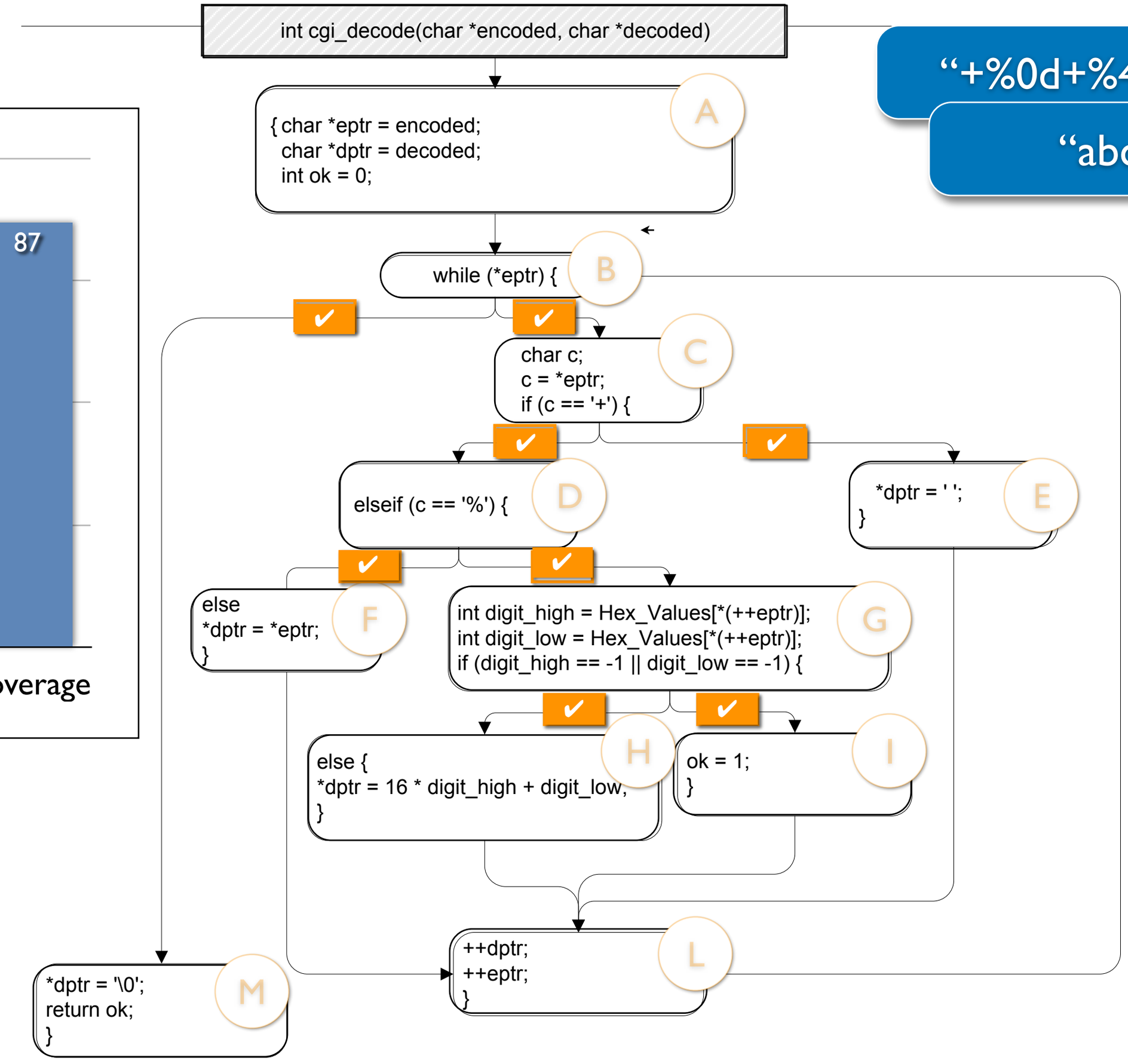

100
75
50
25
0
87
Coverage


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
False
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%0d+%4j"

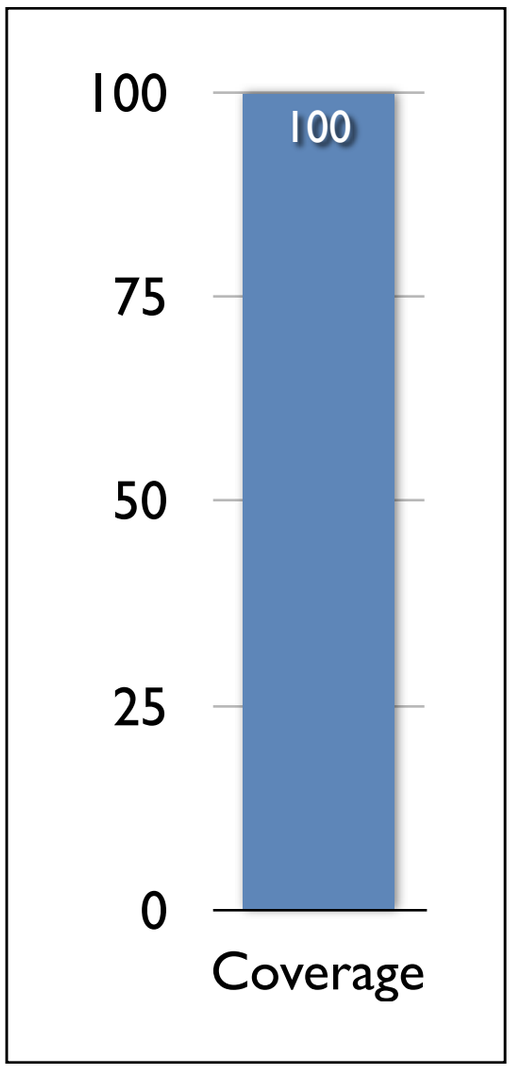
"abc"

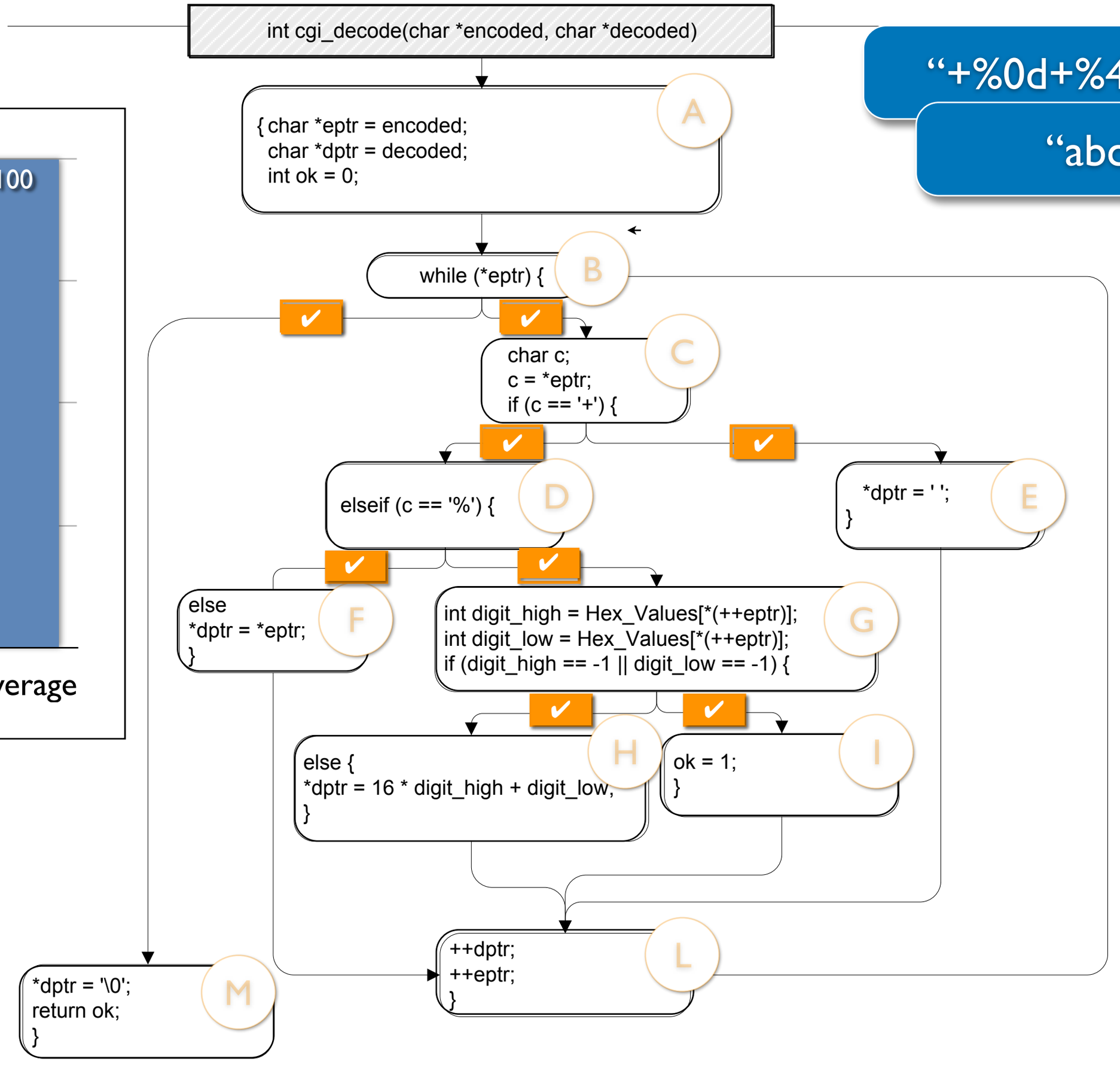

100
75
50
25
0
87
Coverage


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%0d+%4j"

"abc"


100
75
50
25
0
100
Coverage


int cgi_decode(char *encoded, char *decoded)
{ char *eptr = encoded;
char *dptr = decoded;
int ok = 0;
A
while (*eptr) {
B
char c;
c = *eptr;
if (c == '+') {
C
elseif (c == '%') {
D
*dptr = ' ';
}
E
else
*dptr = *eptr;
}
F
int digit_high = Hex_Values[*(++eptr)];
int digit_low = Hex_Values[*(++eptr)];
if (digit_high == -1 || digit_low == -1) {
G
else {
*dptr = 16 * digit_high + digit_low;
}
H
ok = 1;
}
I
++dptr;
++eptr;
}
L
*dptr = '\0';
return ok;
}
M

"+%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.

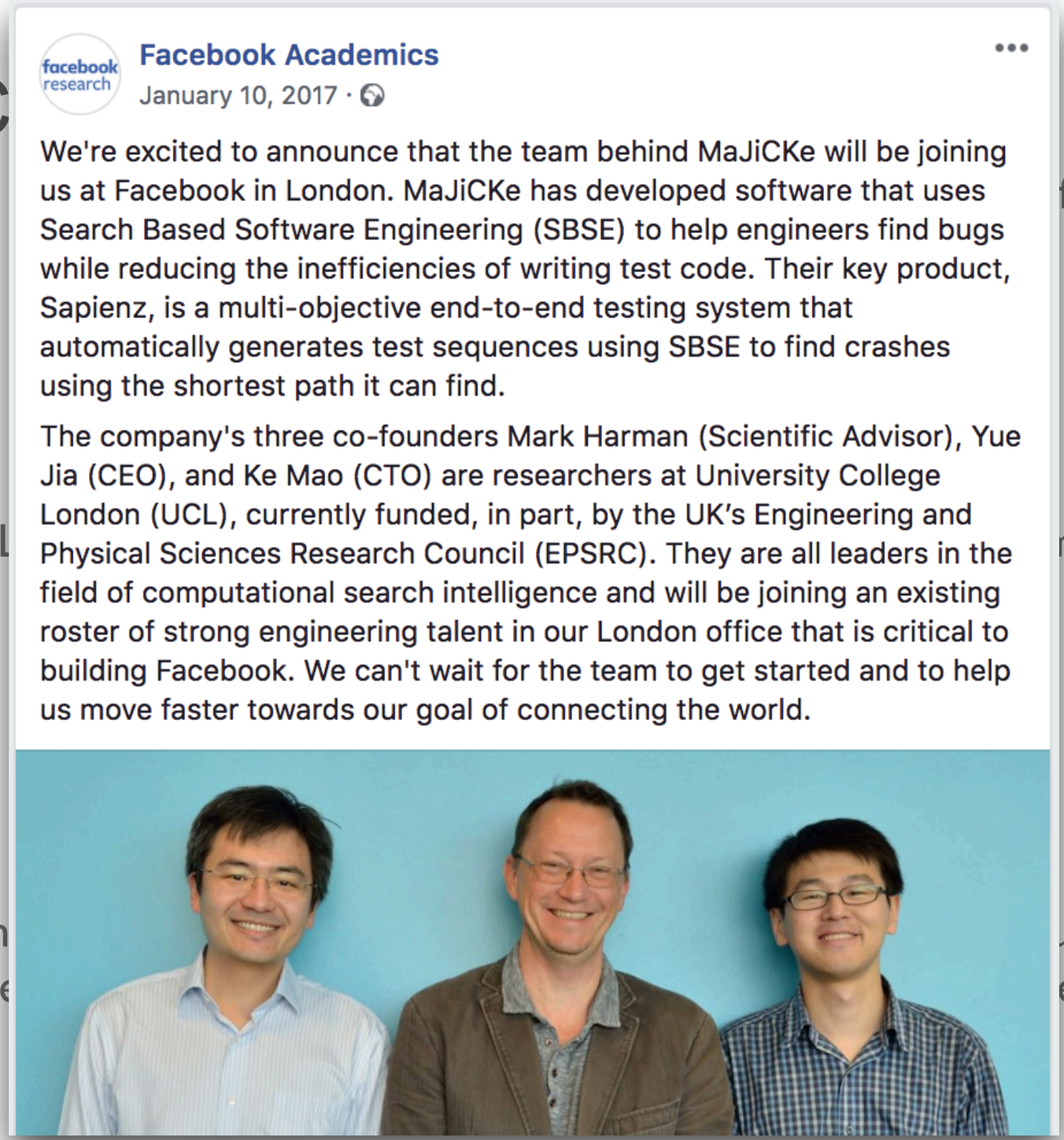
Search-based Testing (SBST)

C g

M.B. Cohen, figurable systems in

L. C. Briand, Y. L ns with genetic

K. Derderian ue input output
sequence gene er Journal, 2006.



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

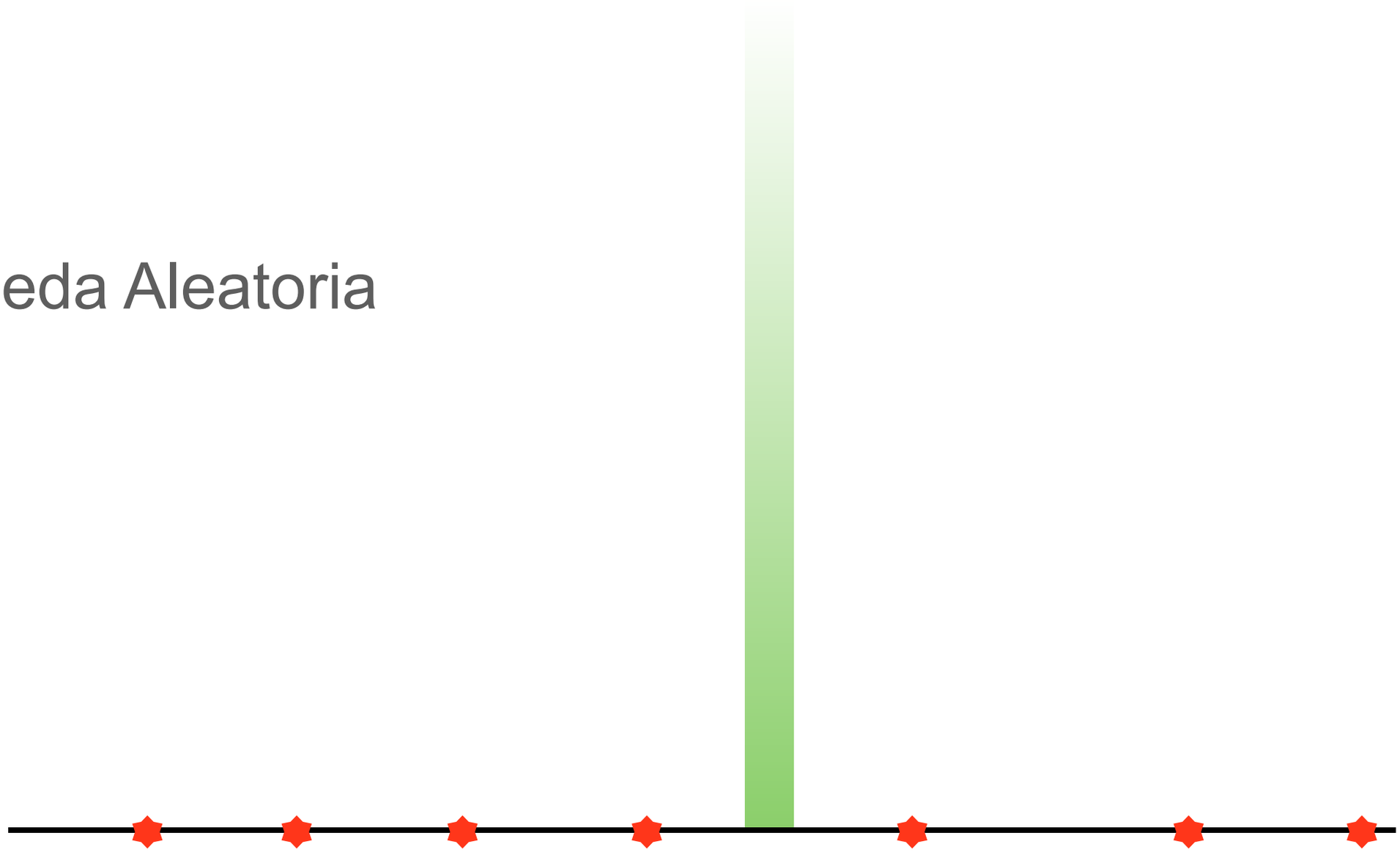
Búsqueda Aleatoria

Entrada

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

Generación Basada en Búsqueda

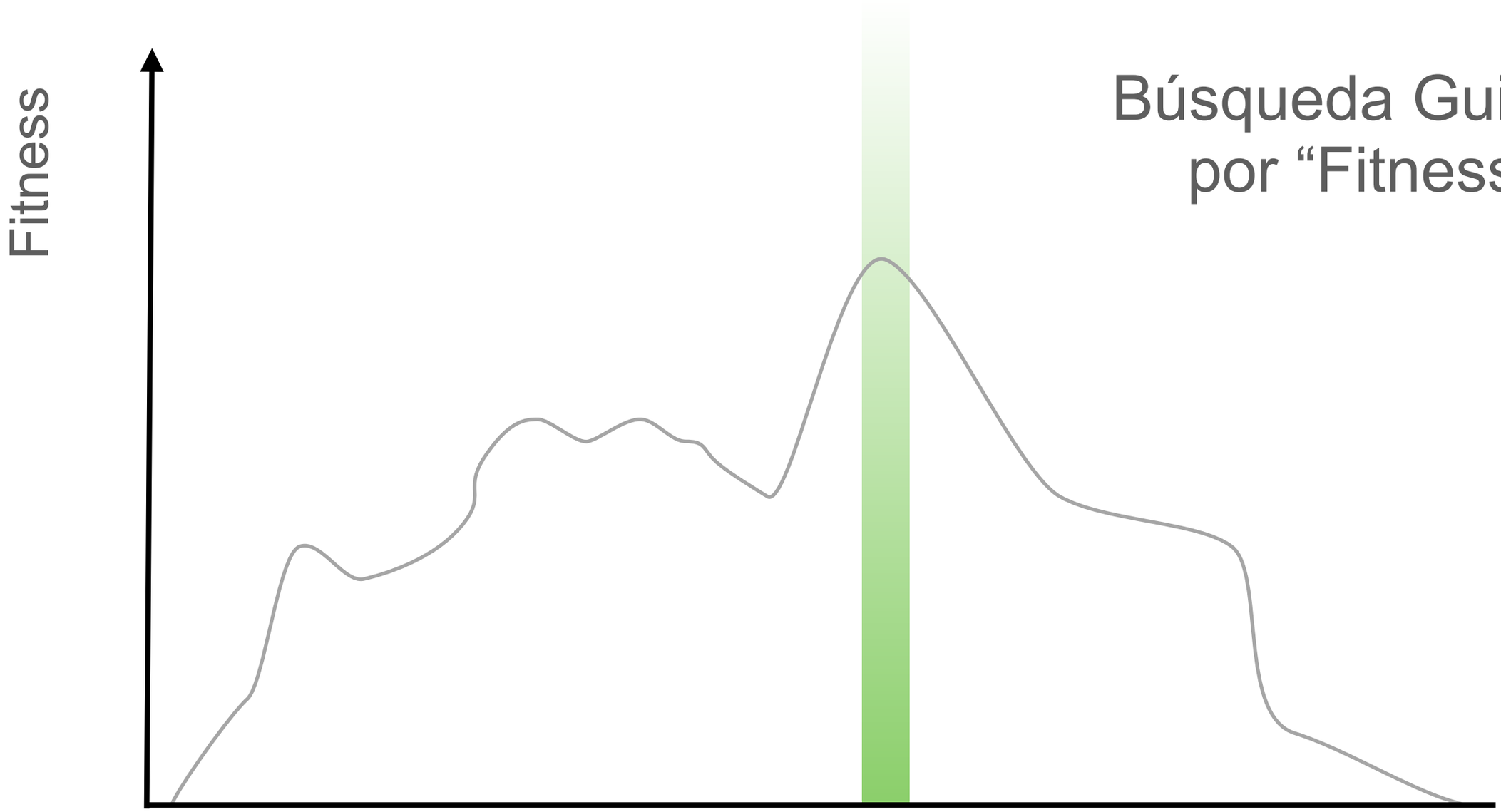
Búsqueda Aleatoria



Entrada

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

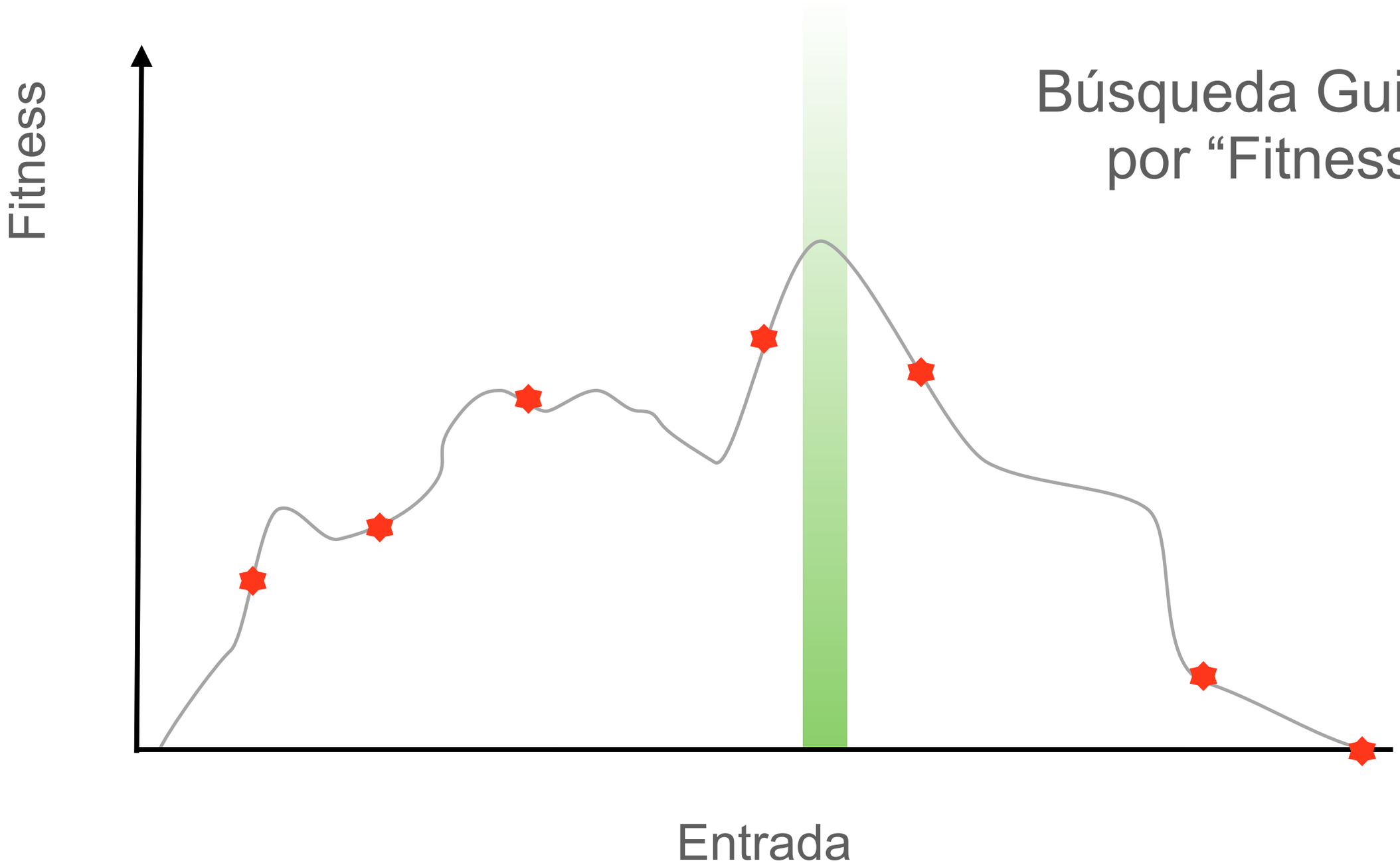
Generación Basada en Búsqueda



Búsqueda Guiada
por “Fitness”

G. Fraser and A. Arcuri, “Whole Test Suite Generation”, TSE 2013.

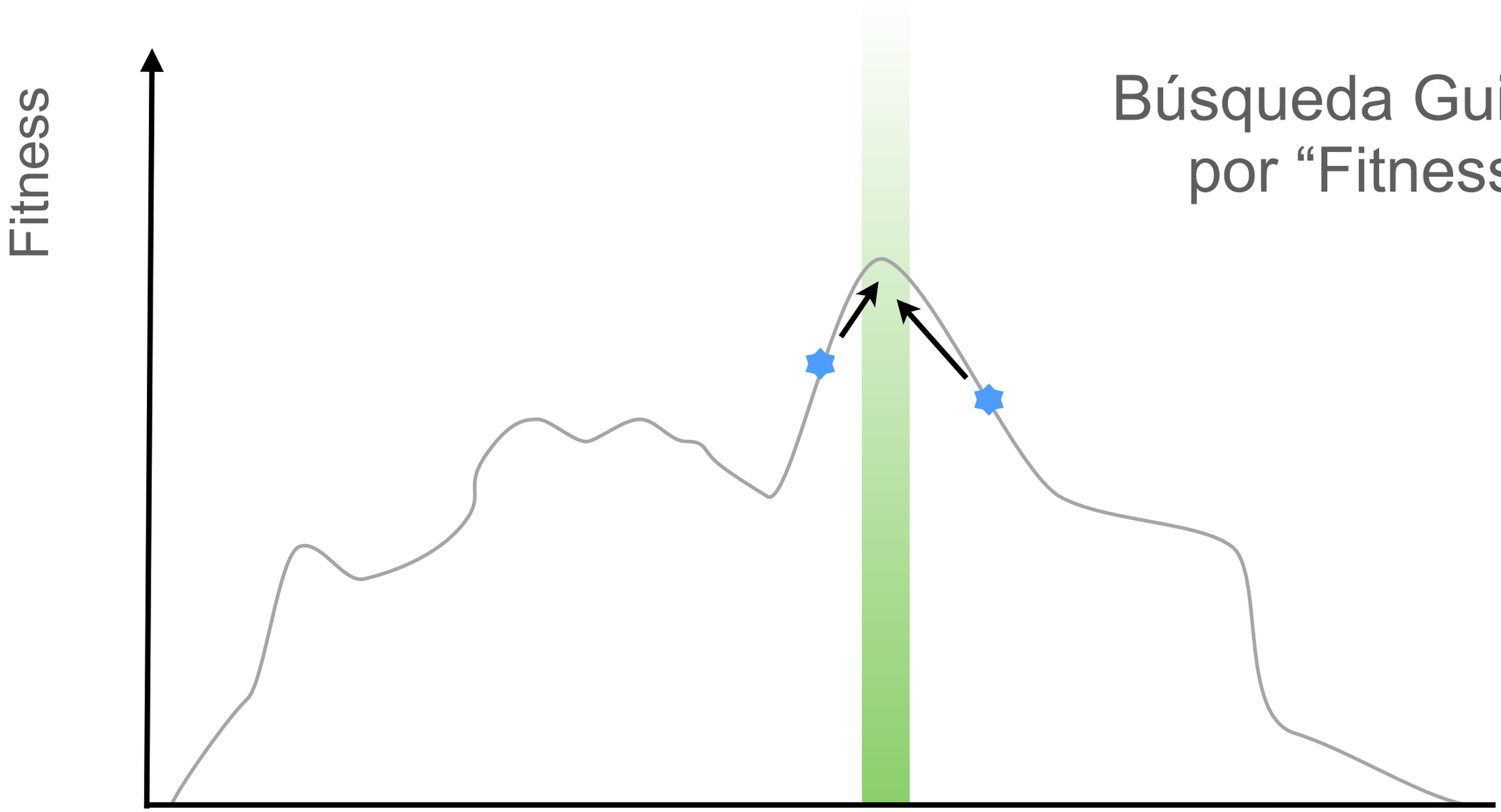
Generación Basada en Búsqueda



Búsqueda Guiada por “Fitness”

G. Fraser and A. Arcuri, “Whole Test Suite Generation”, TSE 2013.

Generación Basada en Búsqueda

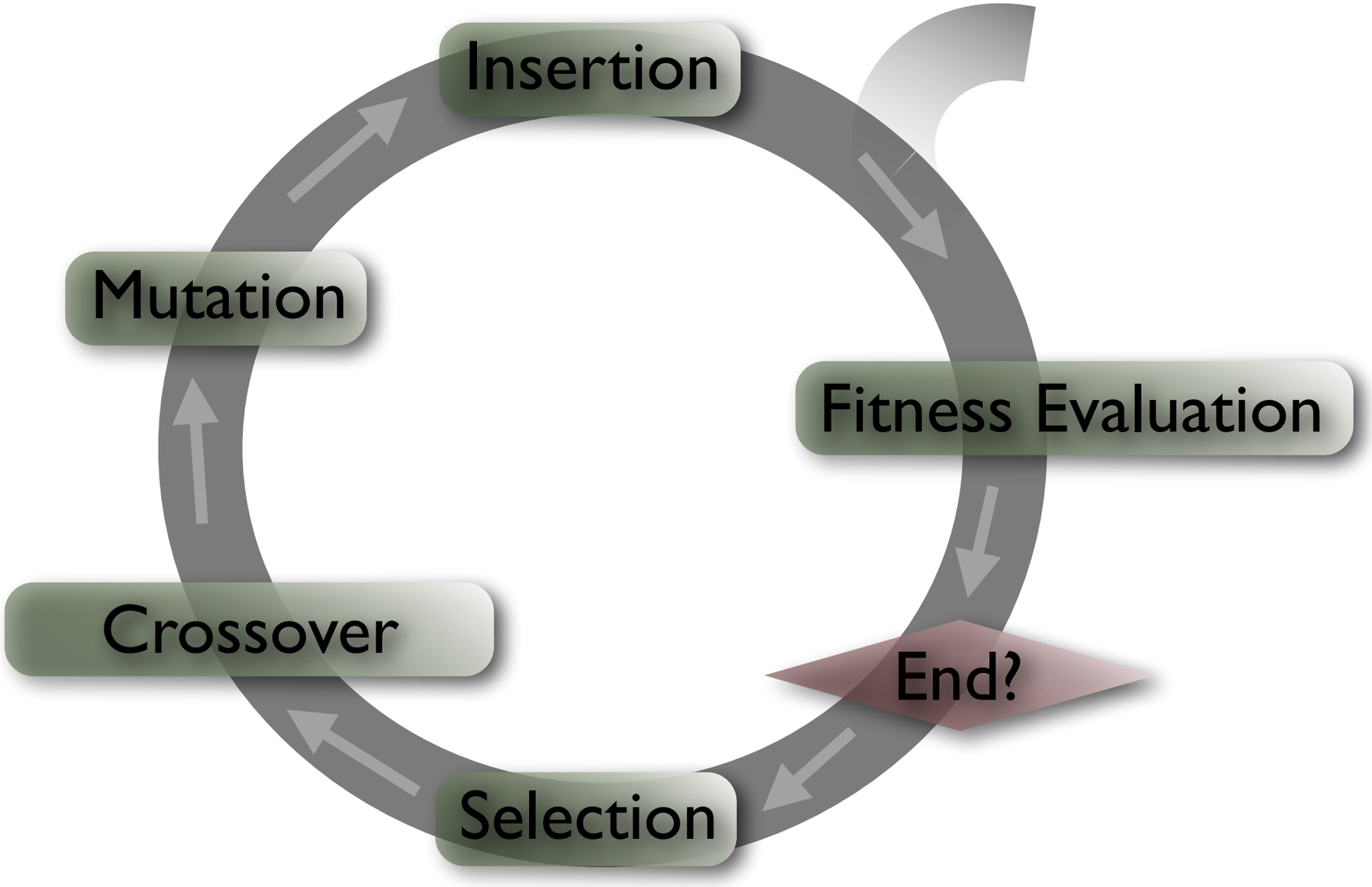


Búsqueda Guiada por “Fitness”

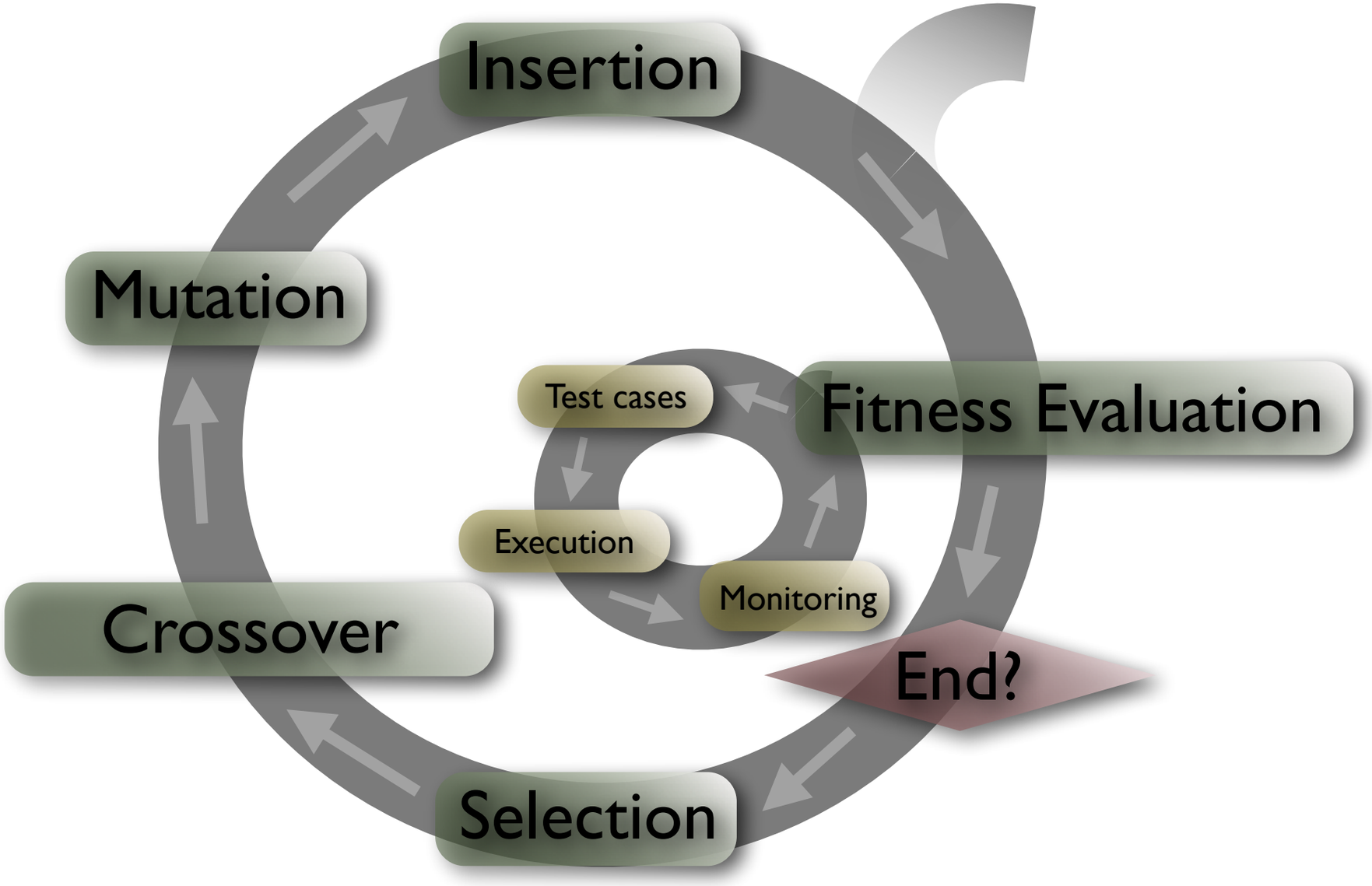
G. Fraser and A. Arcuri, “Whole Test Suite Generation”, TSE 2013.

Algoritmo Genético

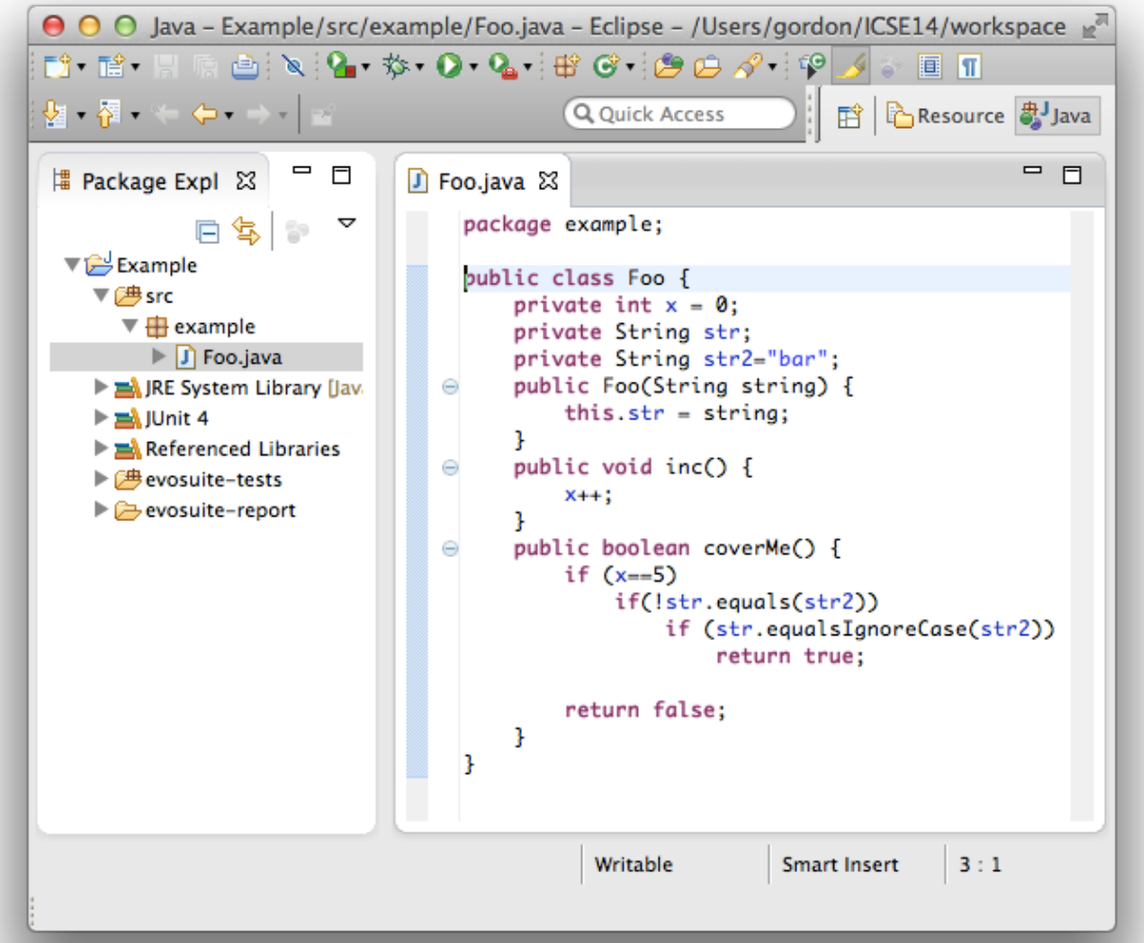
Algoritmo Genético



Algoritmo Genético

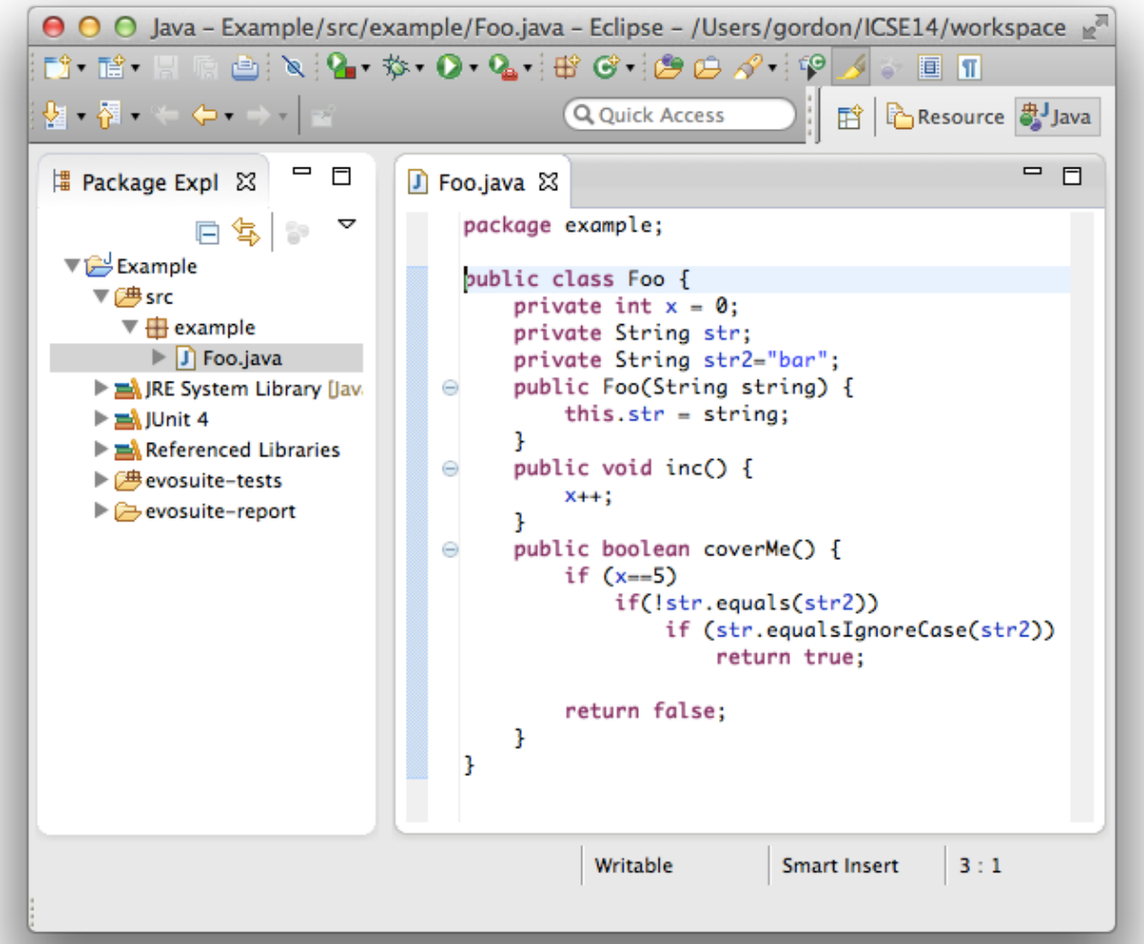


Caso de Uso Básico

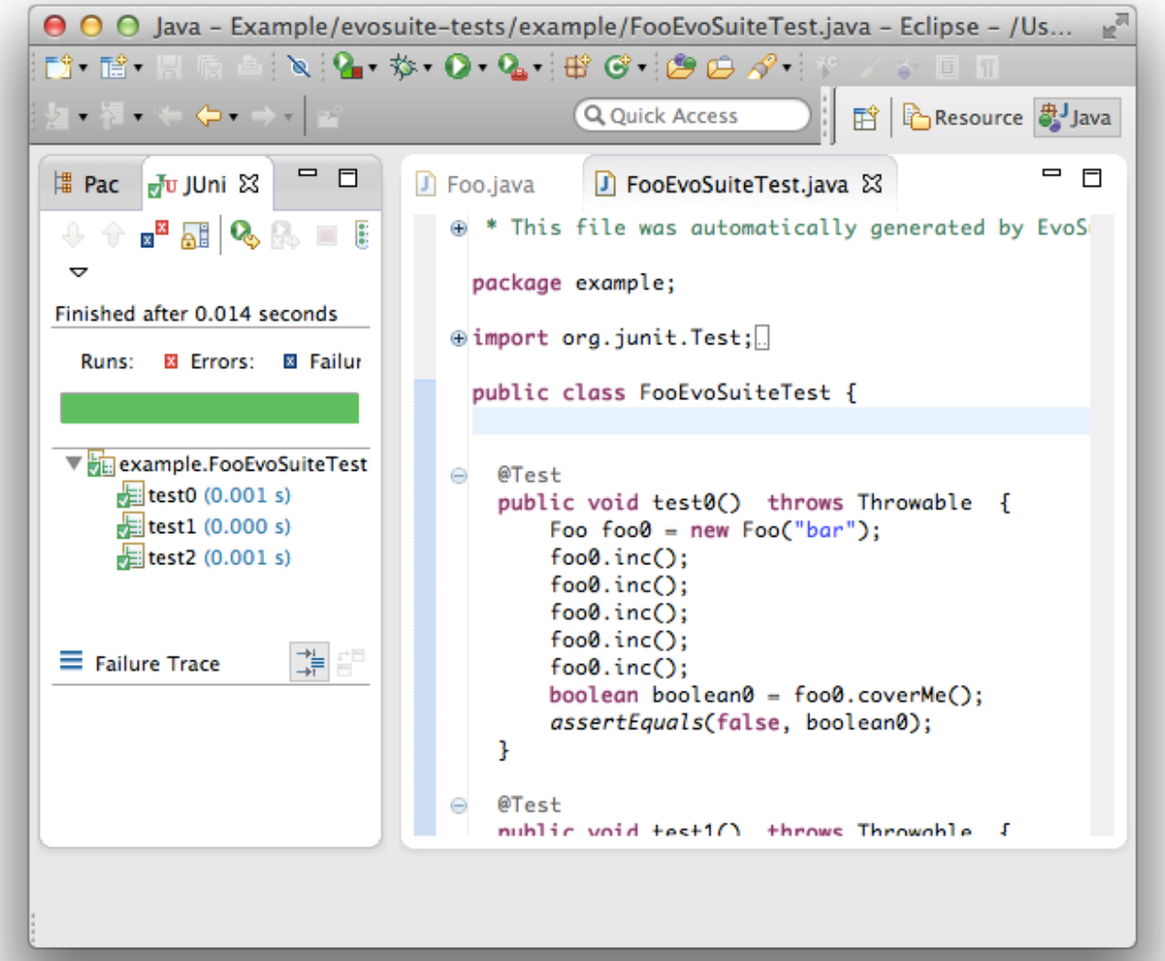


Código Fuente

Caso de Uso Básico



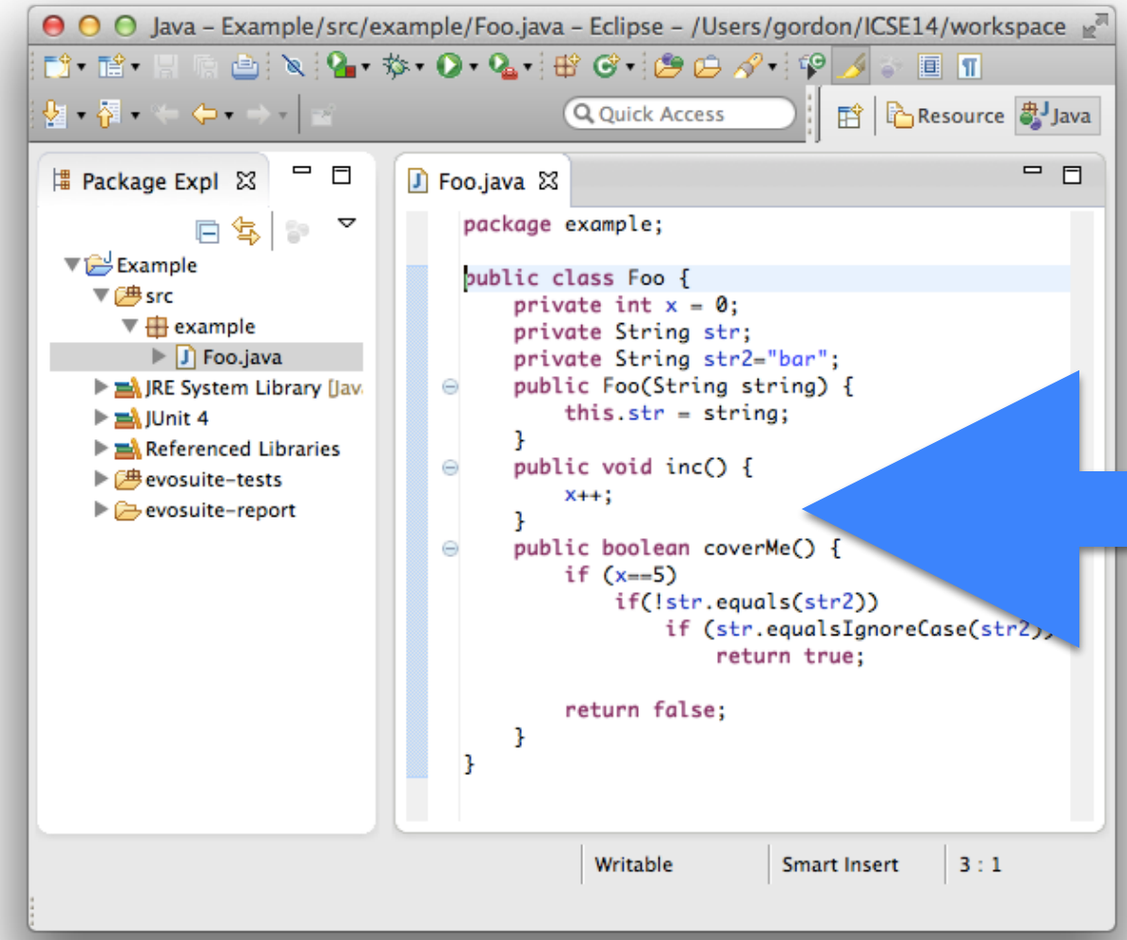
Código Fuente



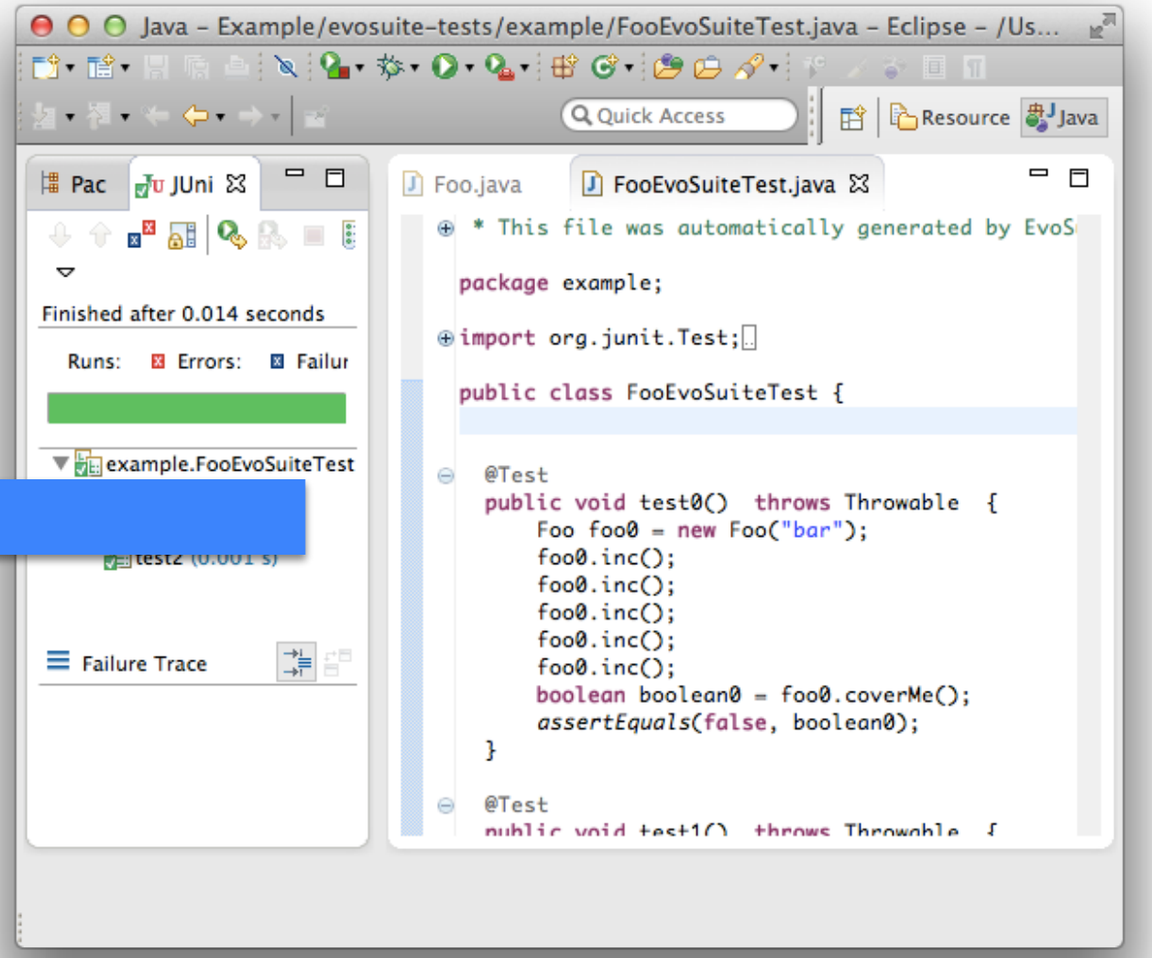
Casos de Prueba

UNIVERSITY OF LEICESTER

Caso de Uso Básico

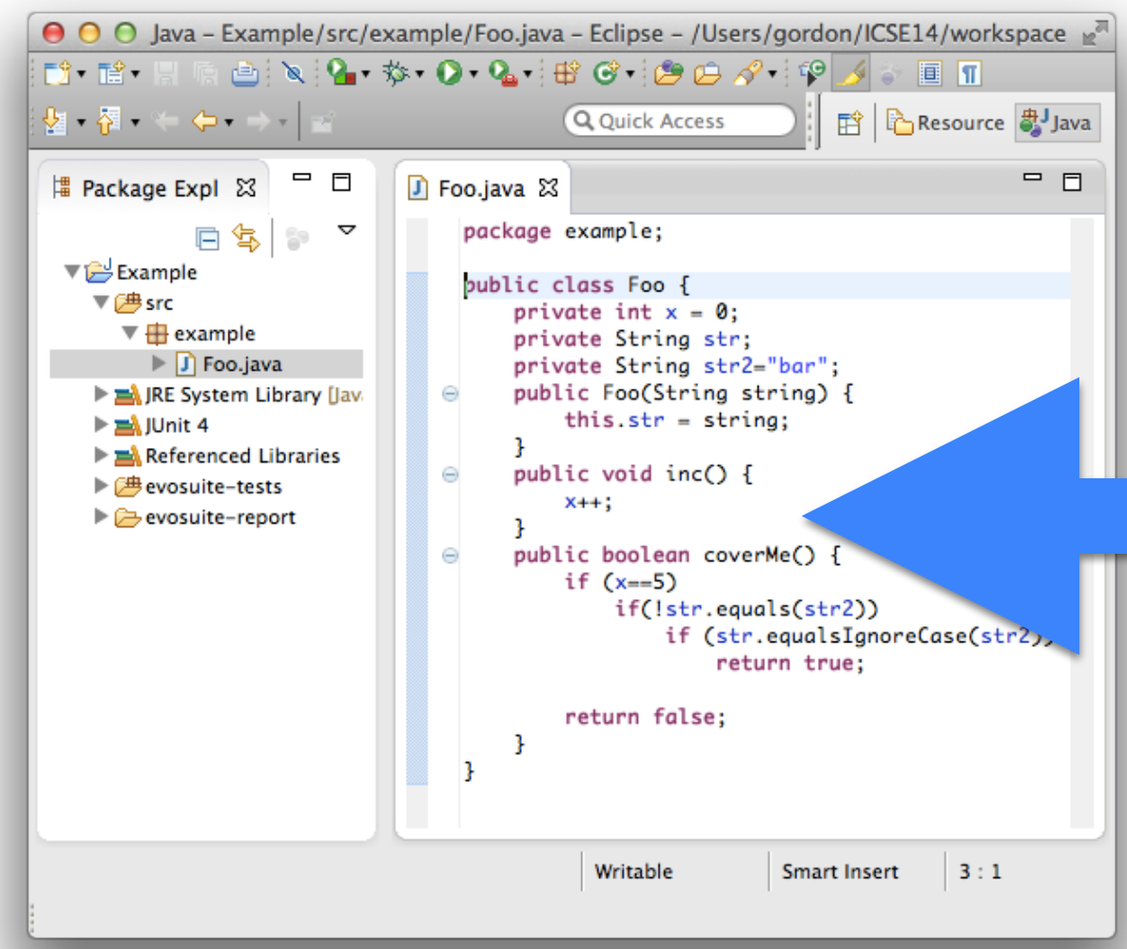


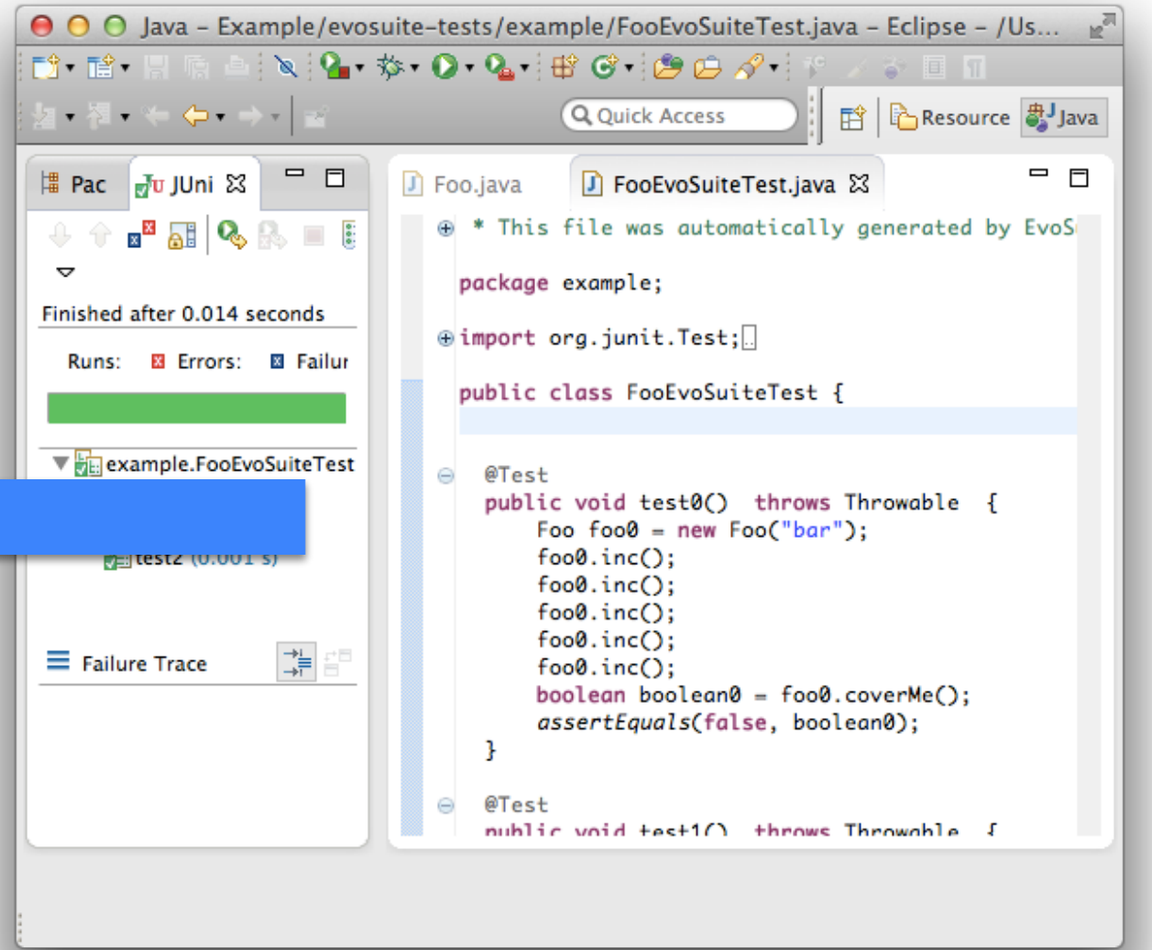
Código Fuente



Casos de Prueba

Caso de Uso Básico





Código Fuente

Casos de Prueba

Generación Automática

UNIVERSITY OF LEICESTER

EVOSUITE

```
@Test
public void test()
{
    int x = 2;
    int y = 2;
    int result = x + y;
    assertEquals(4, result);
}
```

UNIVERSITY OF LEICESTER

EVOSUITE

```
@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);
```

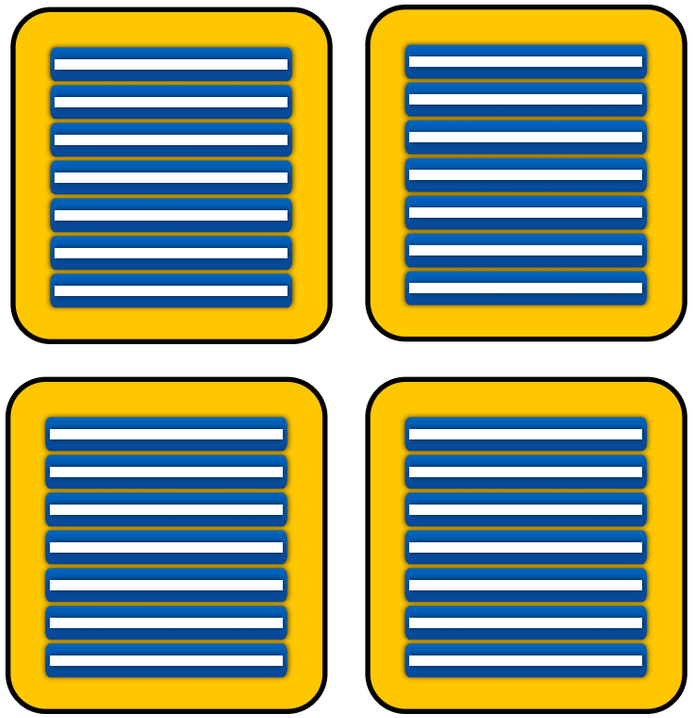
```
}
```

UNIVERSITY OF LEICESTER

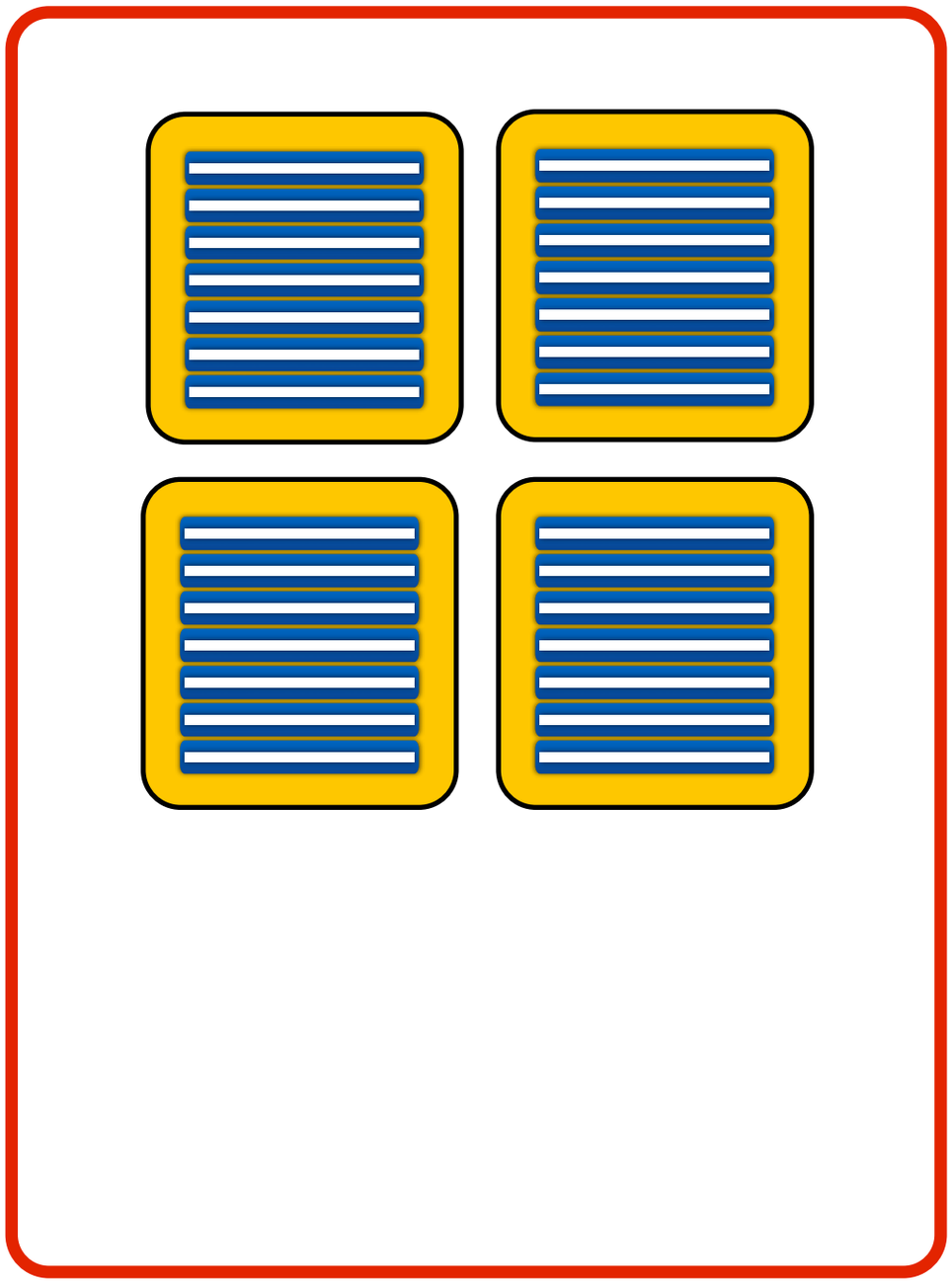
Generación de “Test Suites”

EVOSUITE

Generación de “Test Suites”

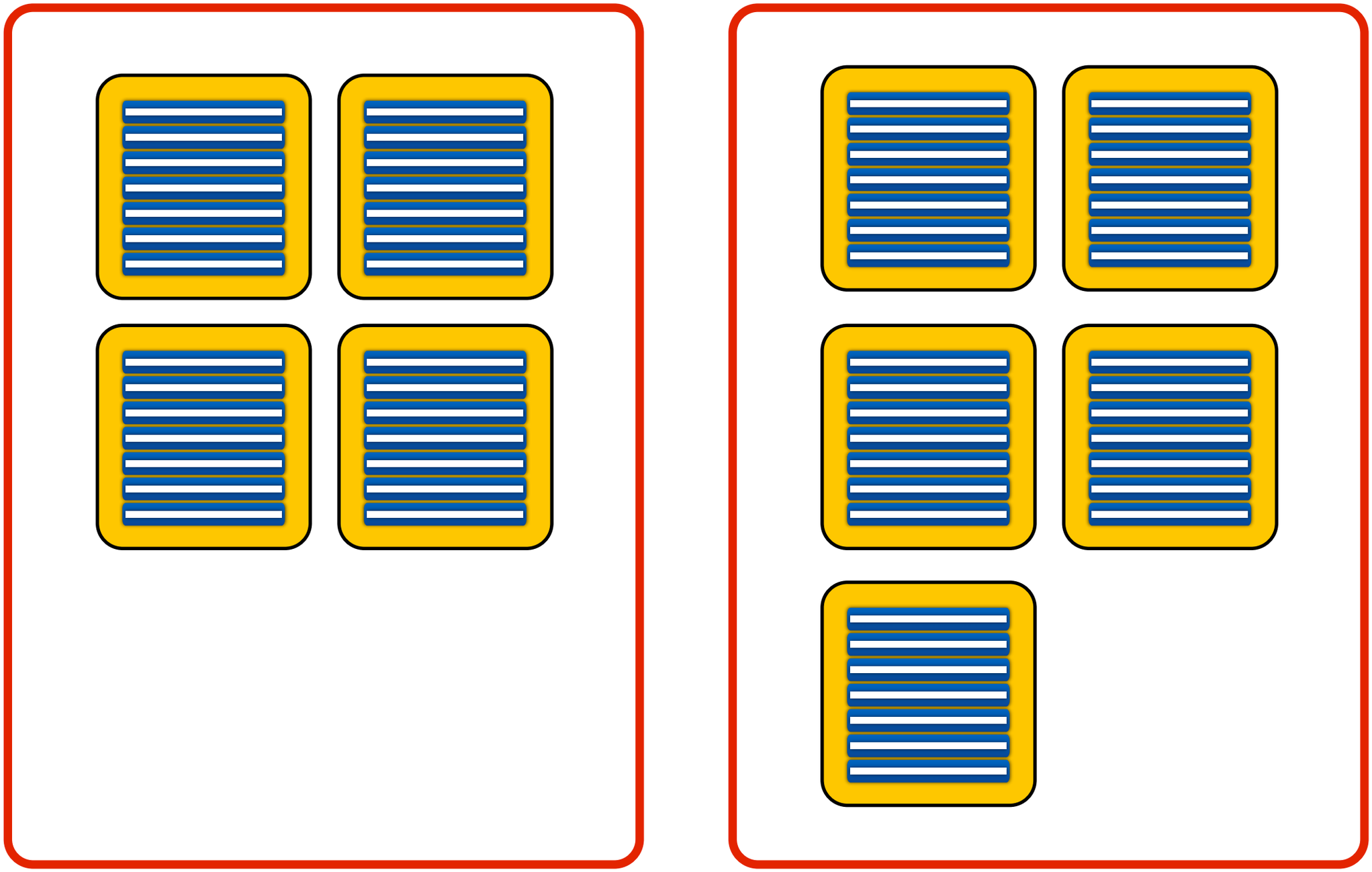


Generación de “Test Suites”

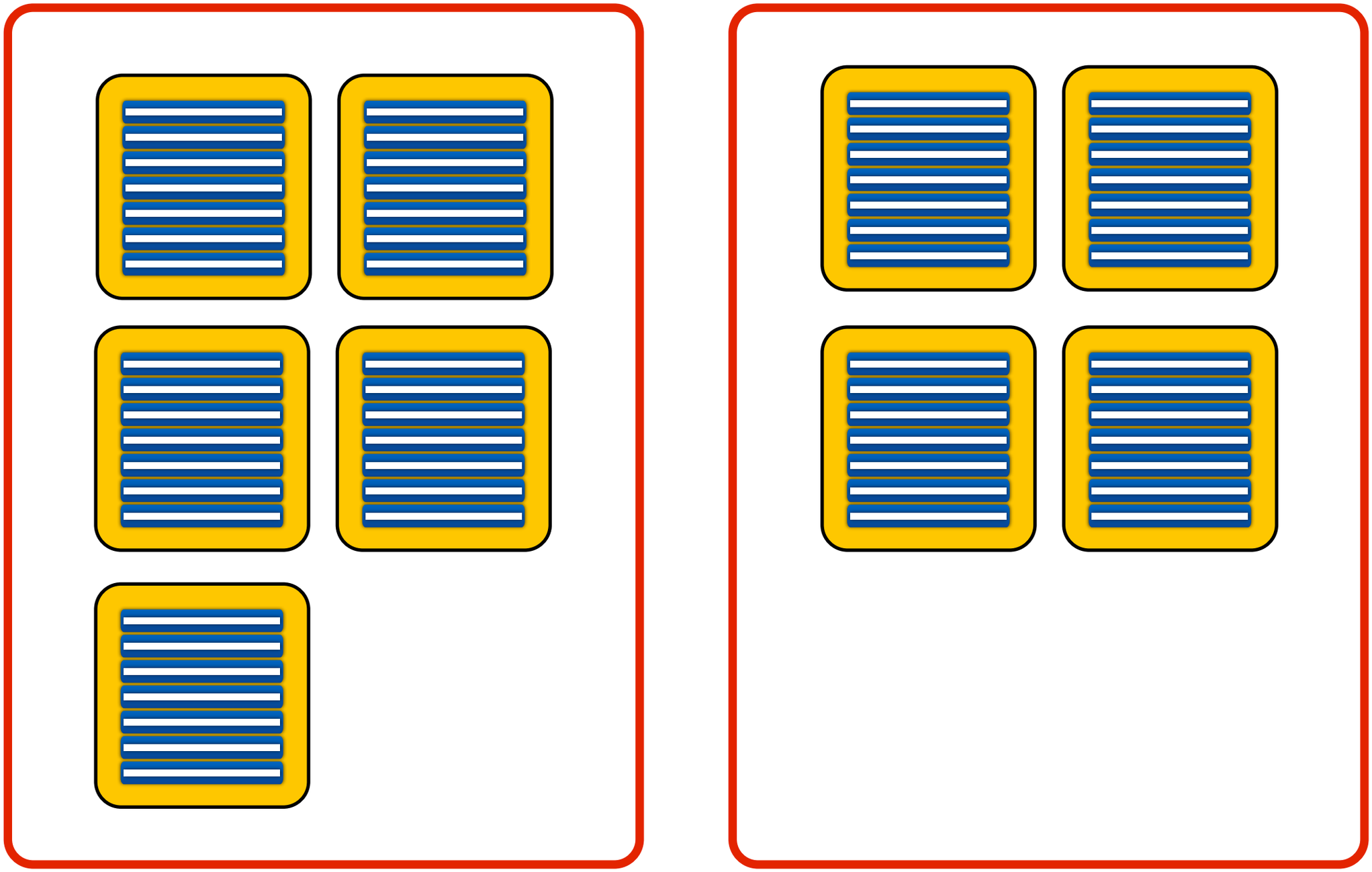


EVOSUITE

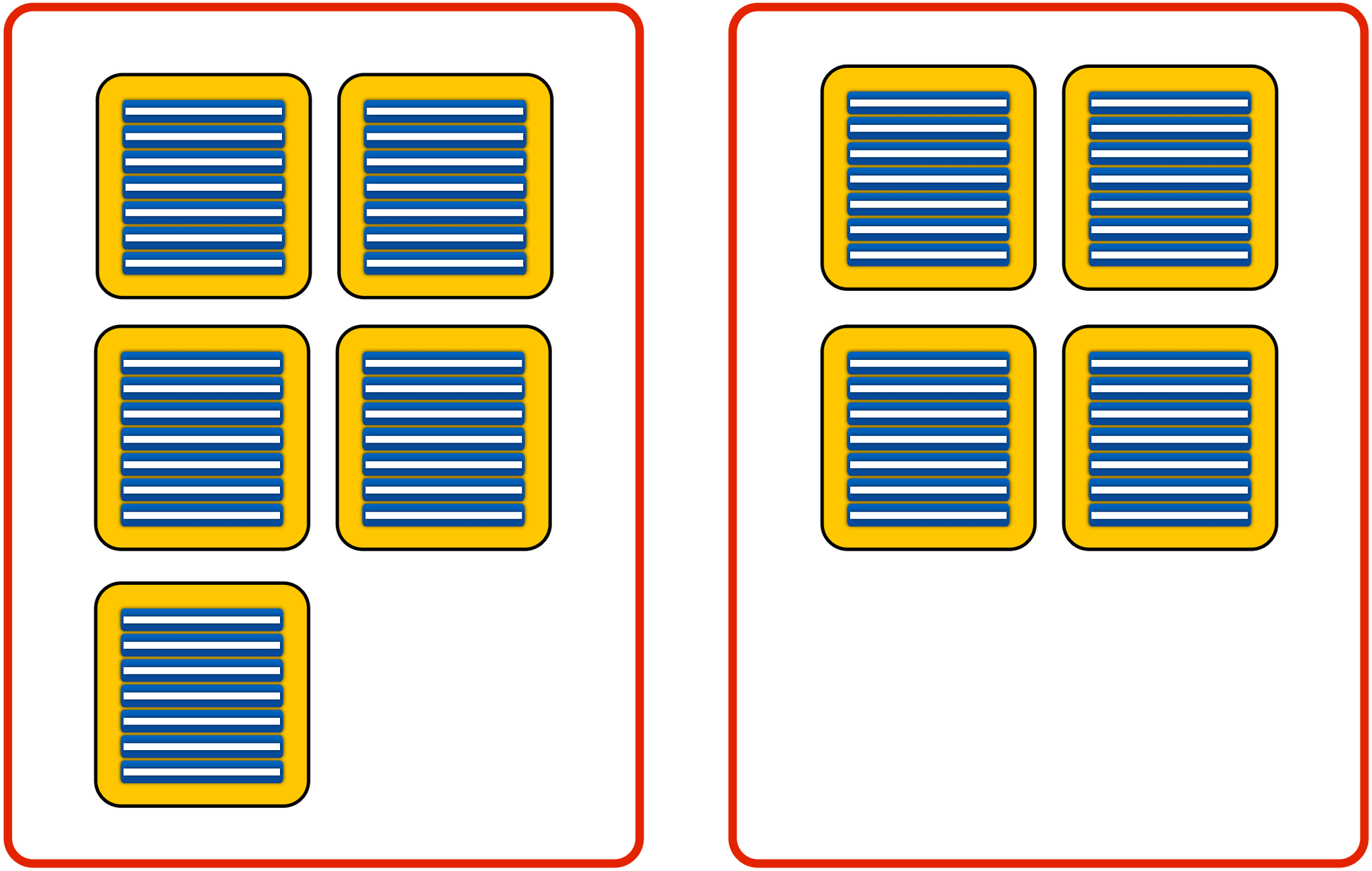
Generación de “Test Suites”



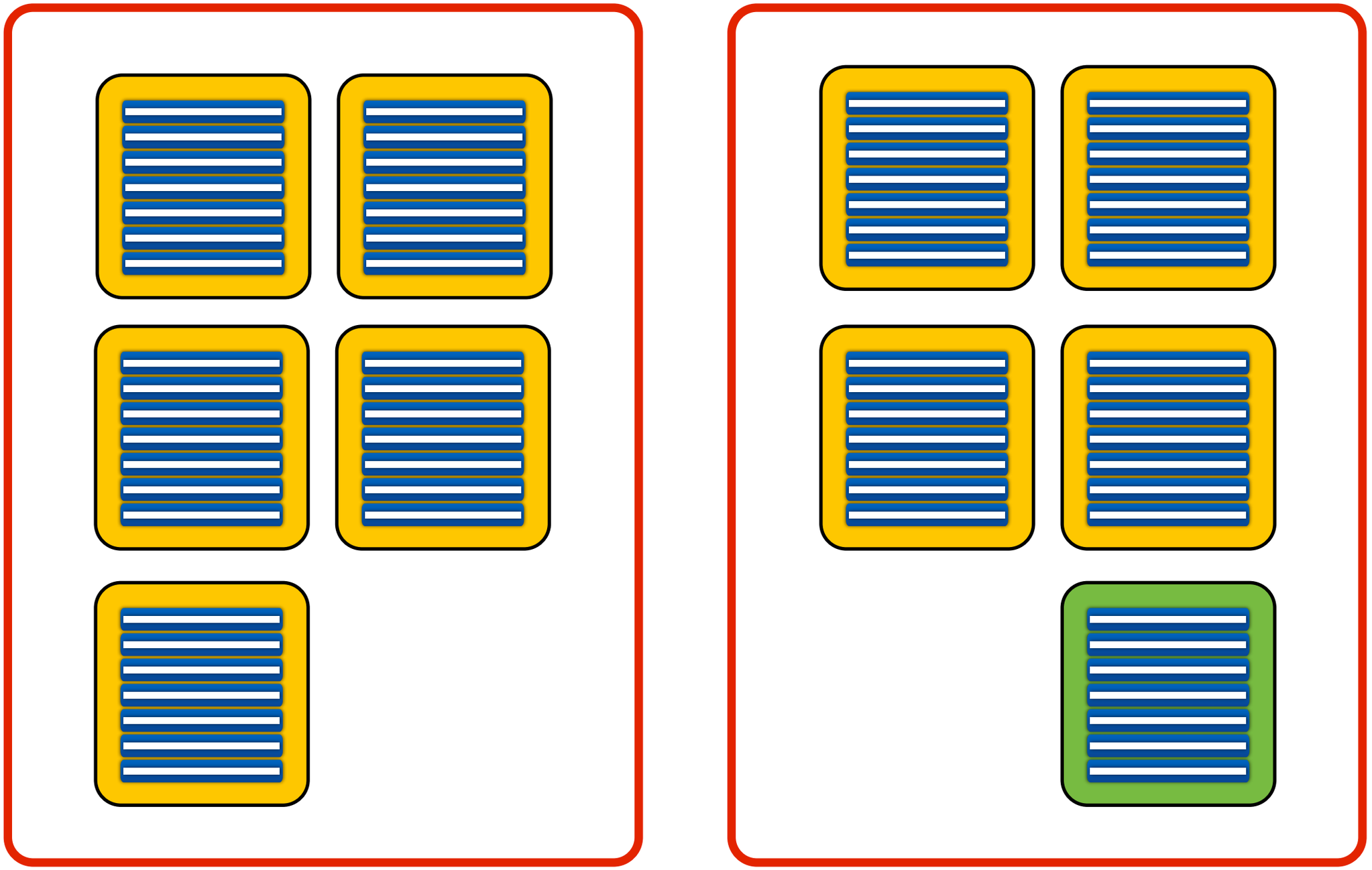
Cruzamiento (Crossover)



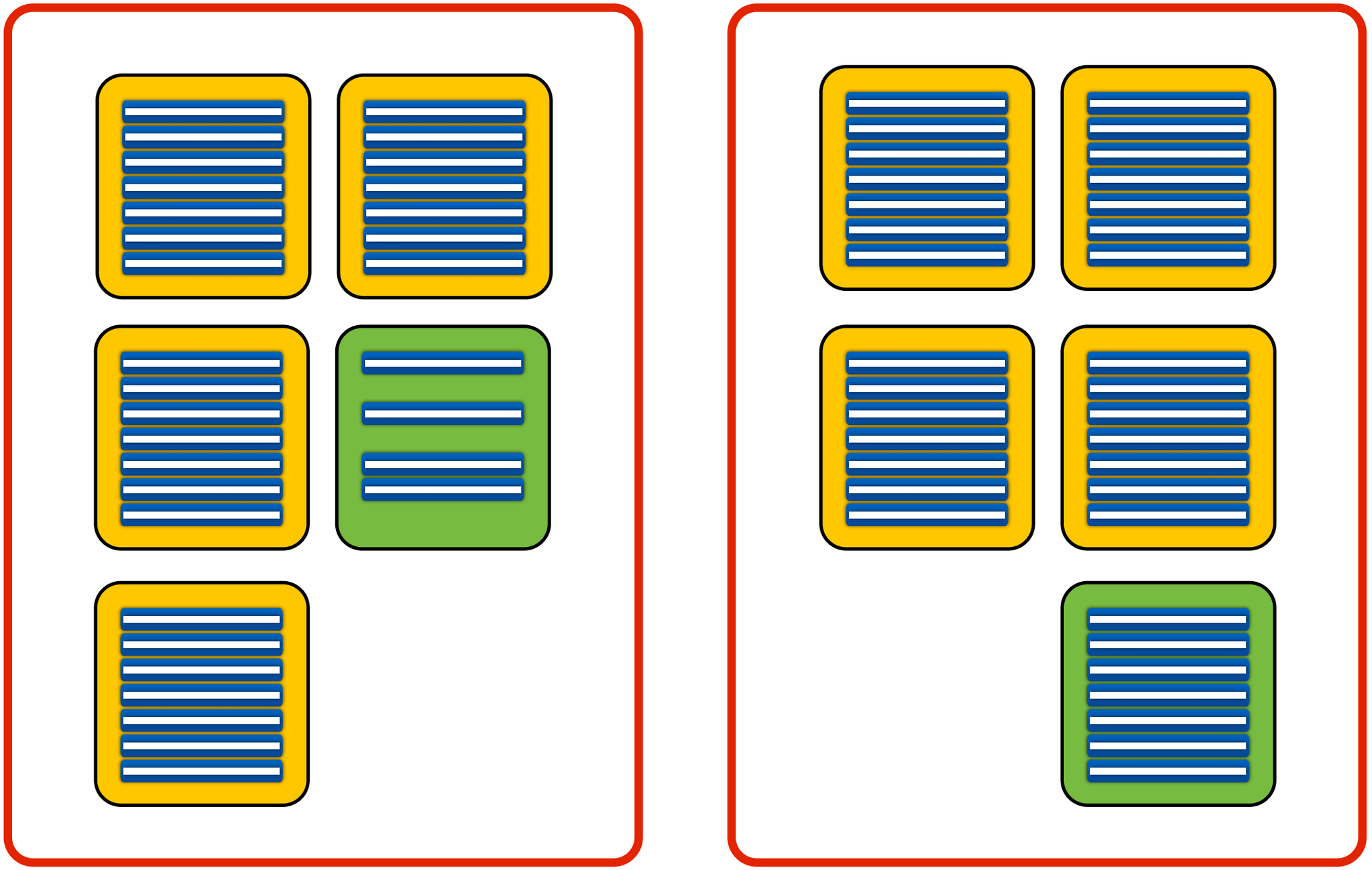
Mutación



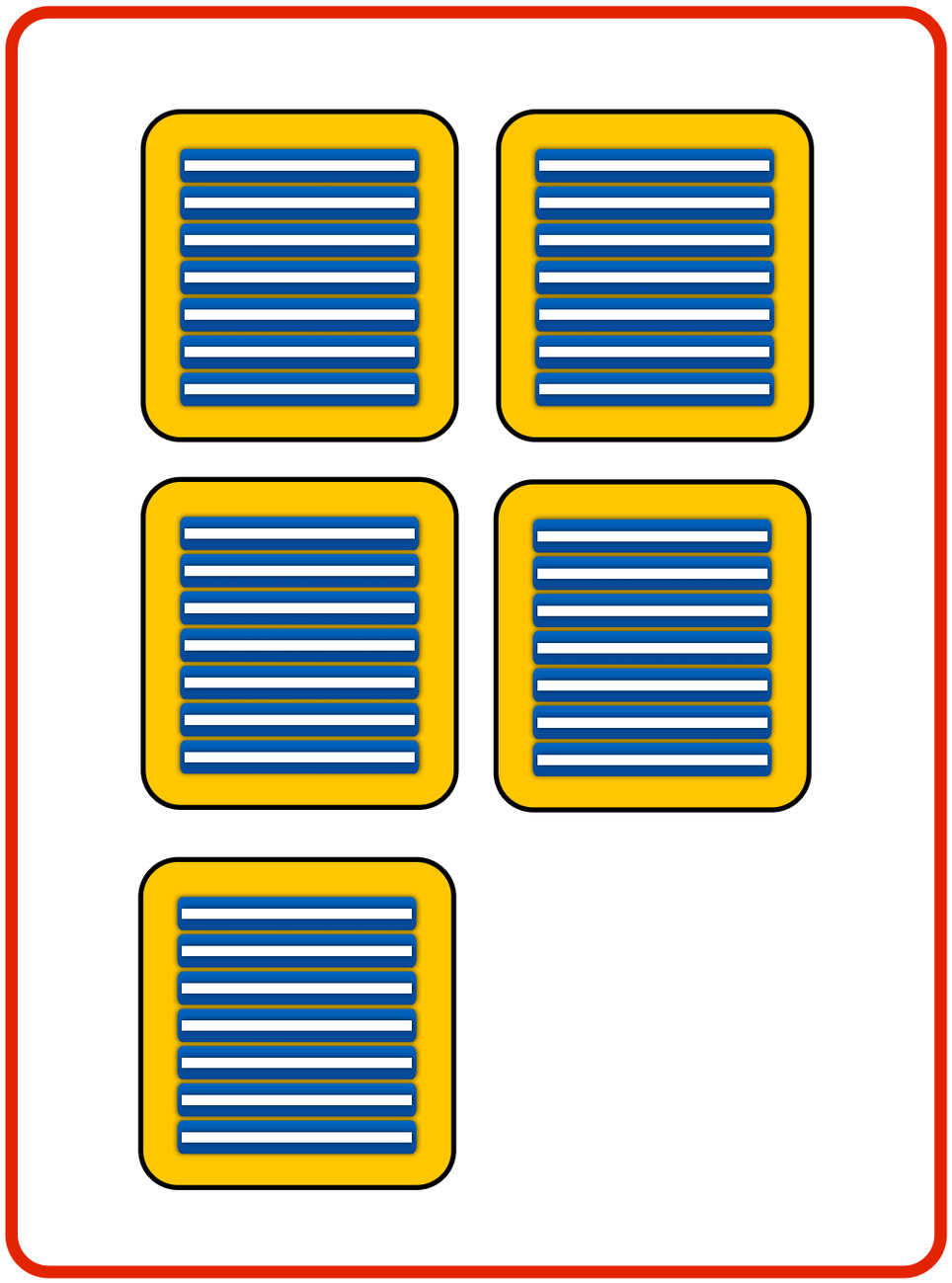
Mutación



Mutación

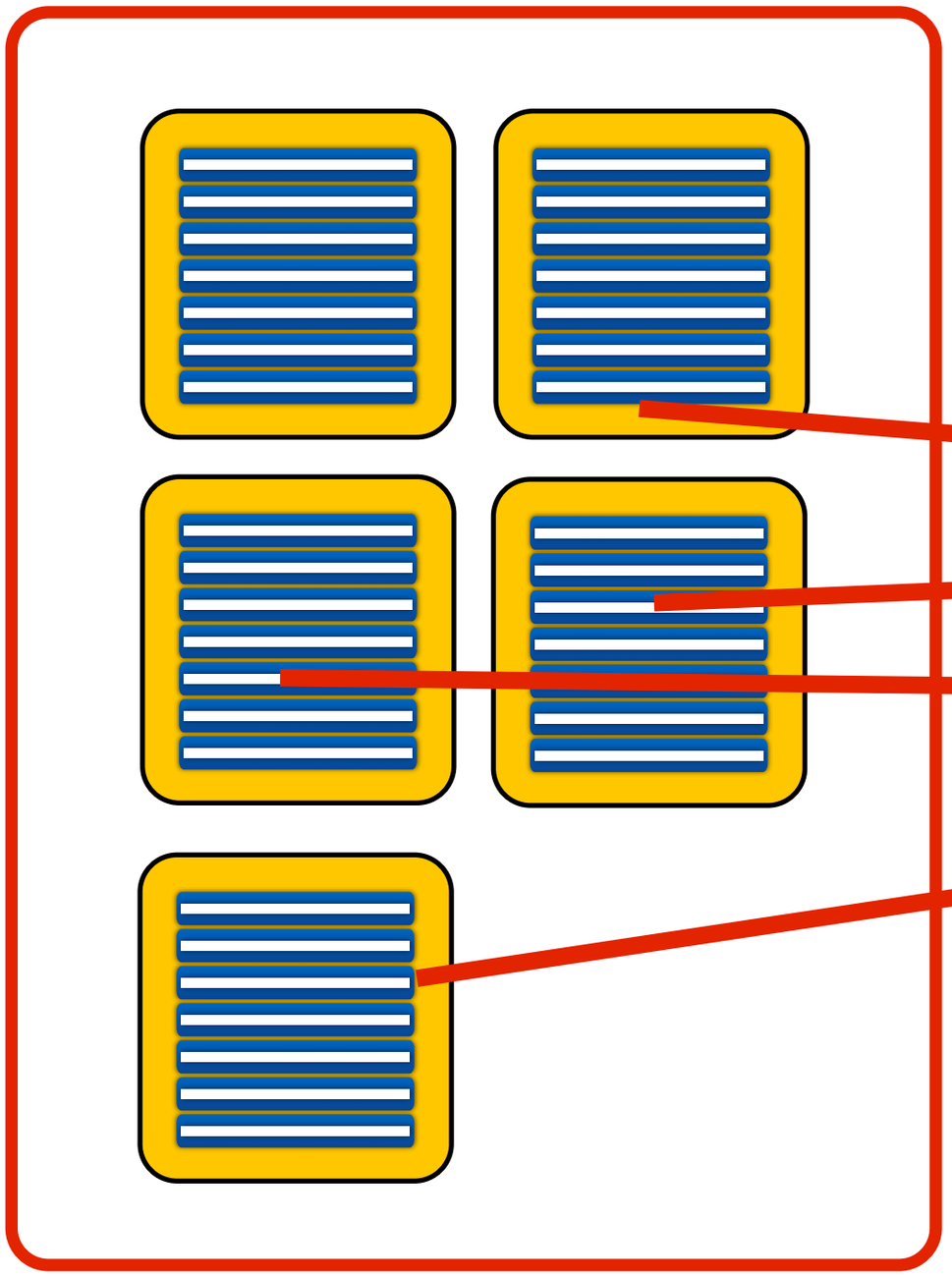


Fitness



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

Fitness



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

EVOSUITE

Código Abierto

http://www.evosuite.org/

https://github.com/evosuite/evosuite

- 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

UNIVERSITY OF LEICESTER

EVOSUITE

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`

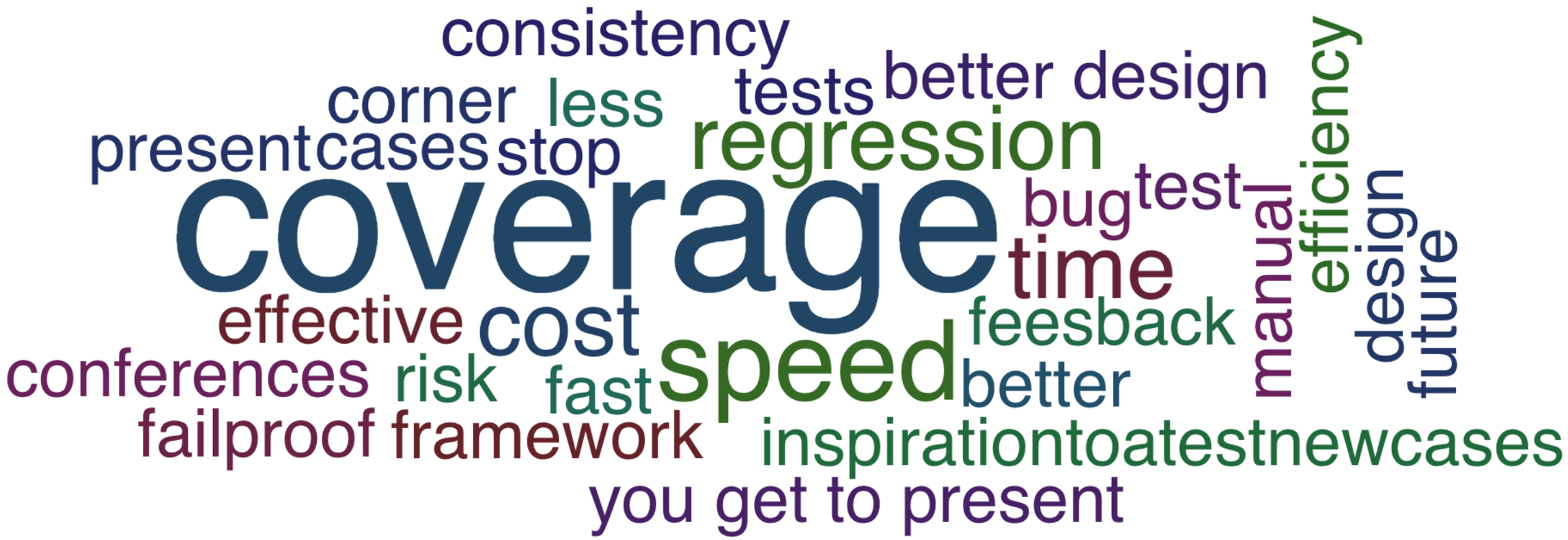
UNIVERSITY OF LEICESTER

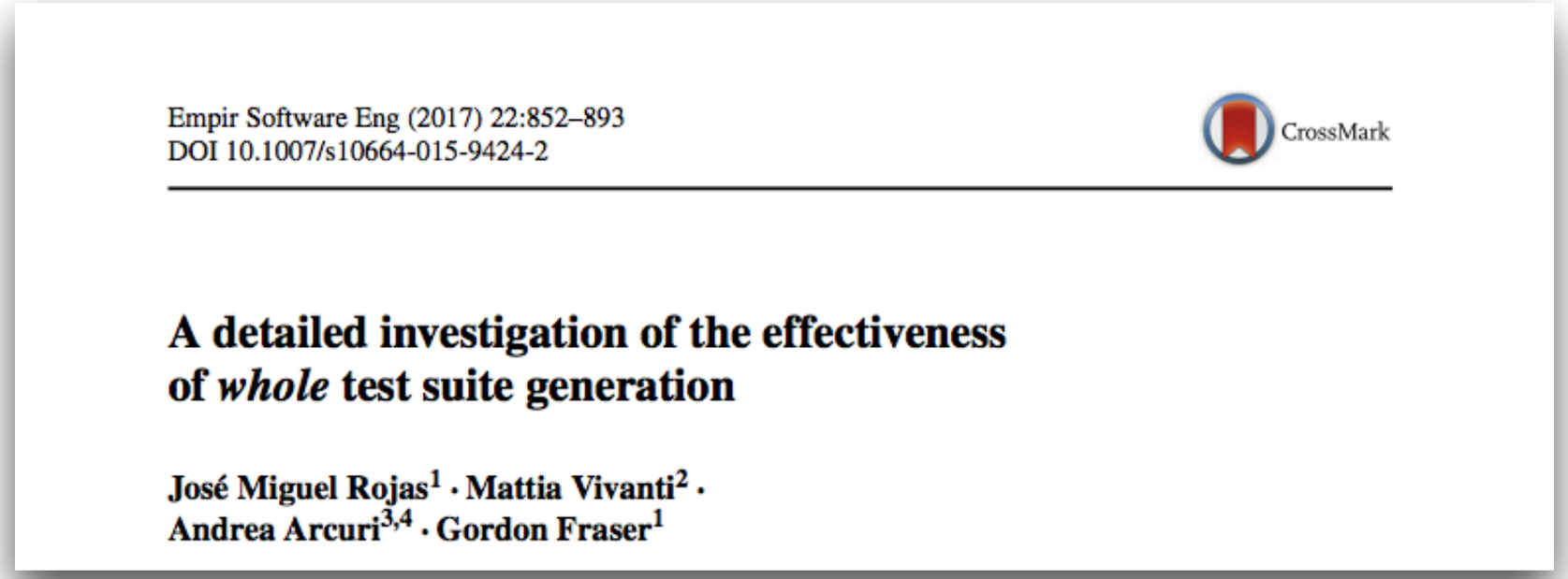
EVOSUITE

DEMO

UNIVERSITY OF LEICESTER

Ventajas?


consistency
corner
less
tests
better design
present
cases
stop
regression
coverage
bug
test
time
manual
effici
ciency
design
future
effective
cost
feesback
conferences
risk
fast
speed
better
failproof
framework
inspirationtoatestnewcases
you get to present



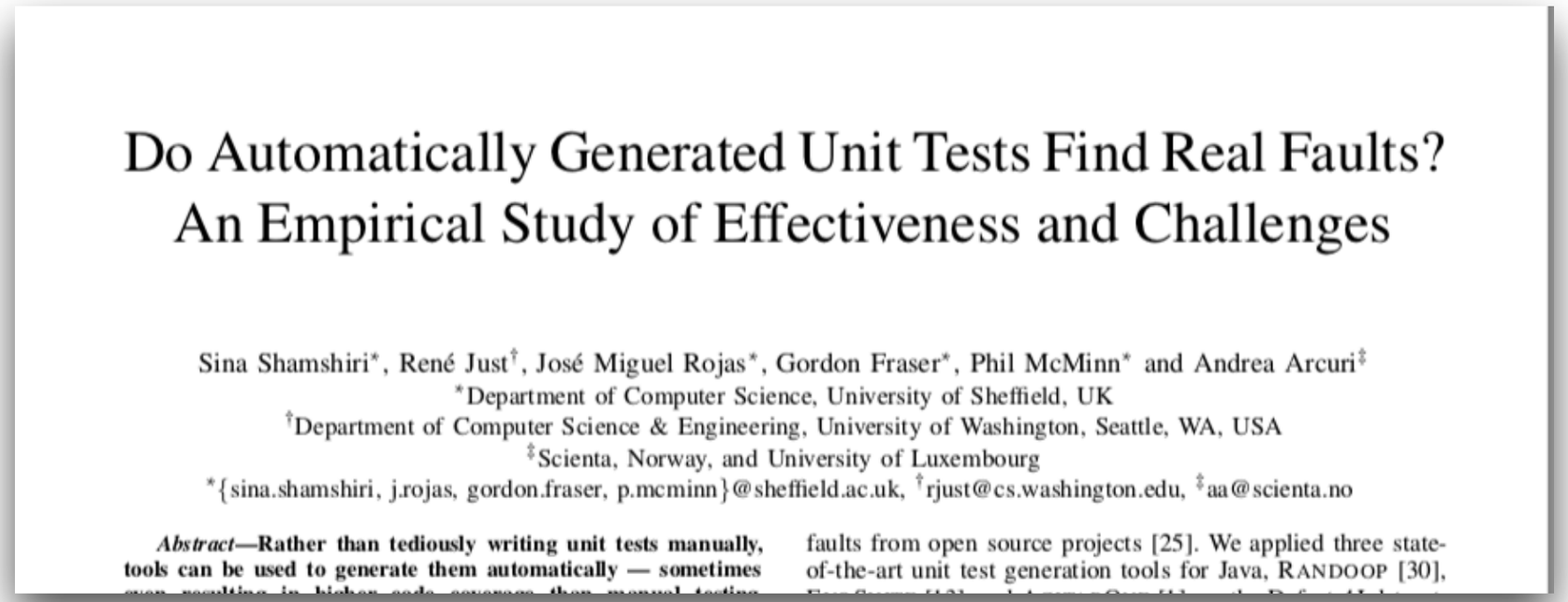
Empir Software Eng (2017) 22:852–893
DOI 10.1007/s10664-015-9424-2

CrossMark

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

José Miguel Rojas[1] · Mattia Vivanti[2] · Andrea Arcuri[3,4] · Gordon Fraser[1]

Code Coverage (EMSE 2017)



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

Sina Shamshiri*, René Just†, José Miguel Rojas*, Gordon Fraser*, Phil McMinn* and Andrea Arcuri‡
*Department of Computer Science, University of Sheffield, UK
†Department of Computer Science & Engineering, University of Washington, Seattle, WA, USA
‡Scienta, Norway, and University of Luxembourg
*{sina.shamshiri, j.rojas, gordon.fraser, p.mcminn}@sheffield.ac.uk, †rjust@cs.washington.edu, ‡aa@scienta.no

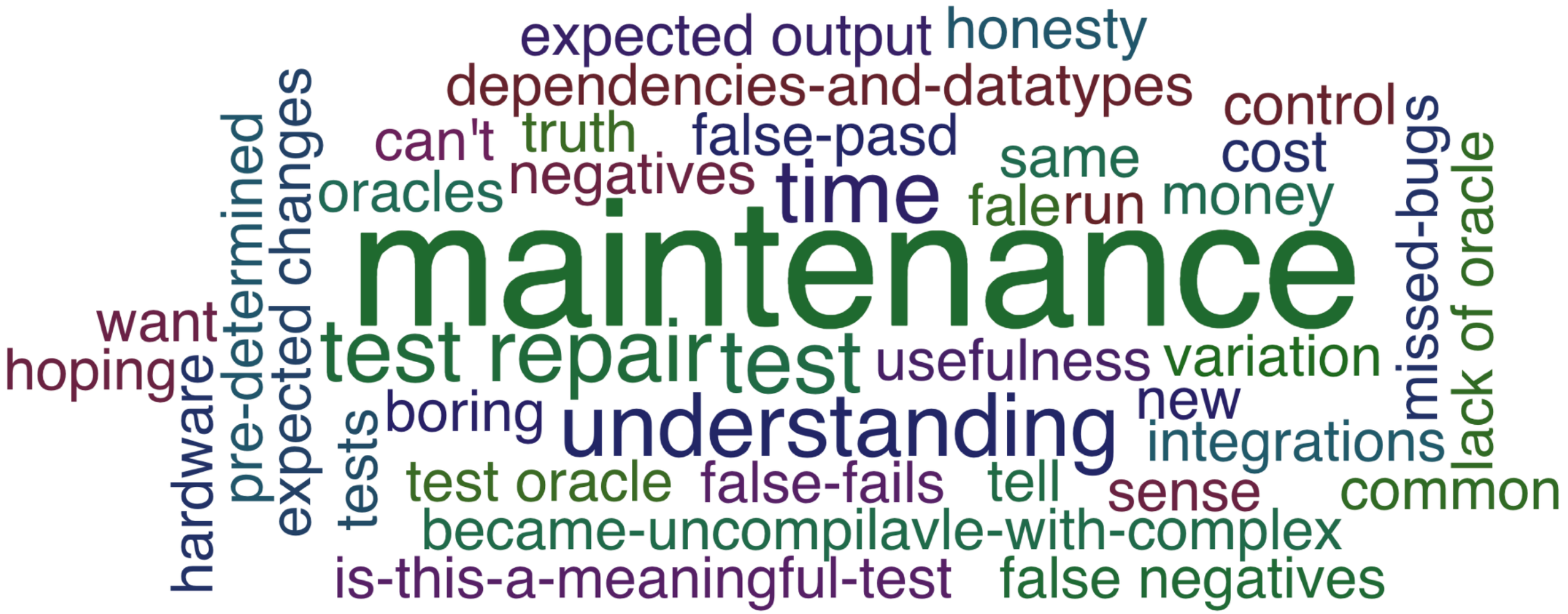
***Abstract*—Rather than tediously writing unit tests manually, tools can be used to generate them automatically — sometimes**

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)

UNIVERSITY OF LEICESTER

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)
       //
    }
}
```

Randoop

```
@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 );
    ...
    ...
}
```

JTeXpert

```
/**
 * 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("YfJOufsFnsYRrFHBpCyLgDWEmhiRPOGISMsFFCYjMOSisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLlYPBYmyihySJwWEVETnqz
BRdlRCMk");
    Double clsUTNumberUtilsP3RP0P2P1OO=-11.825526454314883D;
    Object clsUTNumberUtilsP3RP0P2P1=clsUTNumberUtilsP3RP0P2P1OO;
    String clsUTNumberUtilsP3RP0P2P2OO=new String(">?]$#qxd_ (;jiah+un");
    Object clsUTNumberUtilsP3RP0P2P2=clsUTNumberUtilsP3RP0P2P2OO;
    Short clsUTNumberUtilsP3RP0P2P3OO=1689;
    Object clsUTNumberUtilsP3RP0P2P3=clsUTNumberUtilsP3RP0P2P3OO;
    Object[] clsUTNumberUtilsP3RP0P2=new Object[]
{clsUTNumberUtilsP3RP0P2P1,clsUTNumberUtilsP3RP0P2P2,clsUTNumberUtilsP3RP0P2P3};
    String clsUTNumberUtilsP3RP0R=null;
      clsUTNumberUtilsP3RP0R=String.format(clsUTNumberUtilsP3RP0P1,(Object[])clsUTNumberUtilsP3RP0P2);

assertEquals("YfJOufsFnsYRrFHBpCyLgDWEmhiRPOGISMsFFCYjMOSisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLlYPBYmyihySJwWE
VETnqzBRdlRCMk",clsUTNumberUtilsP3RP0P1.toString());

assertEquals("YfJOufsFnsYRrFHBpCyLgDWEmhiRPOGISMsFFCYjMOSisquxKCuSHIOrrEBbqqwfBtBmhuiUWUSsKUAfloLlYPBYmyihySJwWE
VETnqzBRdlRCMk",clsUTNumberUtilsP3RP0R);
  }
```

Test Readability

"[Developers] read tests […] 77% of the total time they spend in them"

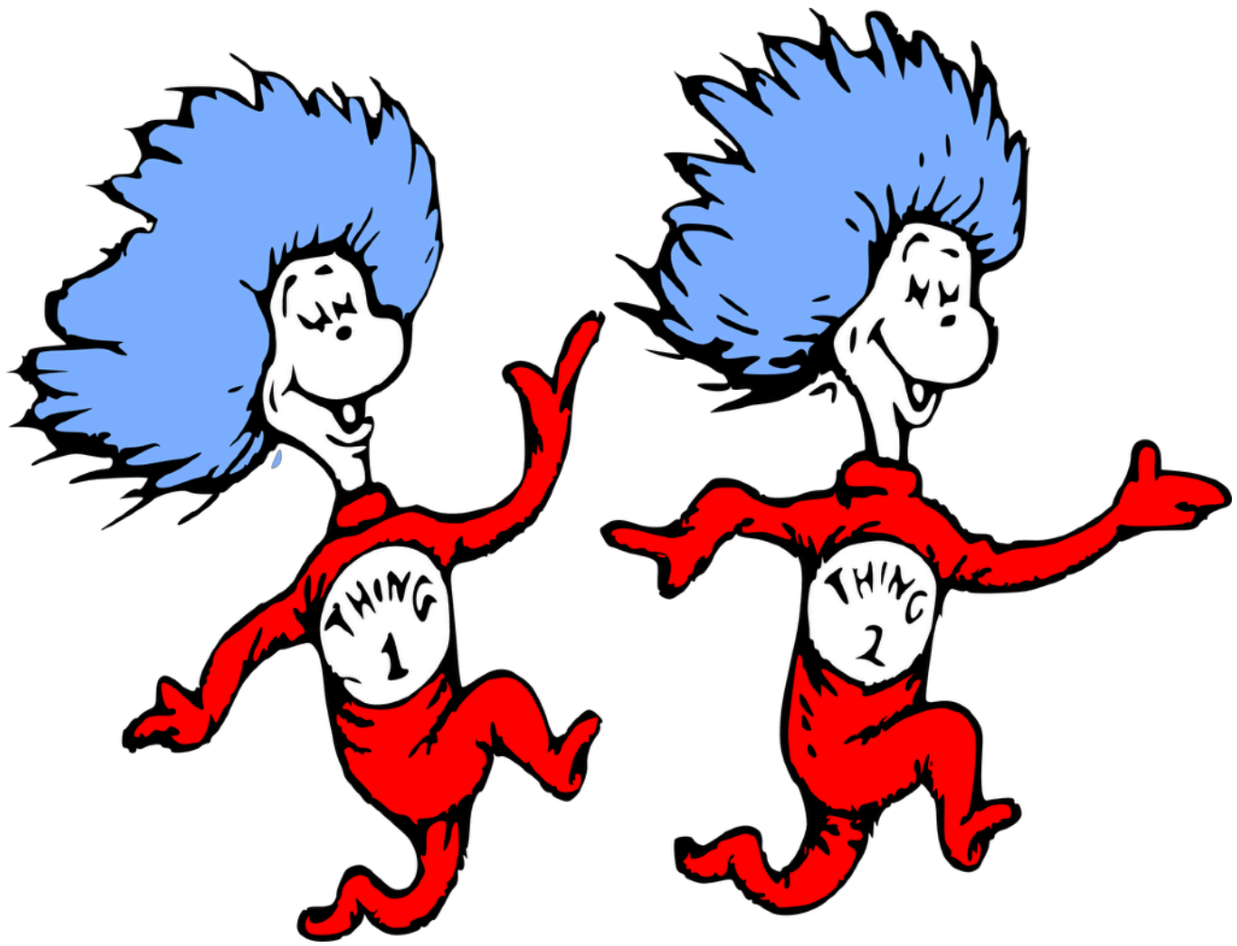
Moritz Beller, Georgios Gousios, Annibale Panichella, and Andy Zaidman, "When, How, and Why Developers (Do Not) Test in Their IDEs". FSE 2015.

Generación de Casos de Prueba con Nombres Descriptivos

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

UNIVERSITY OF LEICESTER

Generación de Casos de Prueba con Nombres Descriptivos



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

UNIVERSITY OF LEICESTER

```
@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

EVOSUITE

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;
   }
}
```

UNIVERSITY OF LEICESTER

EVOSUITE

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

UNIVERSITY OF LEICESTER

Coverage-Based Naming

```
public class ShoppingCart {
   private int total = 0;
   private final static int MAX = 1000;

   public boolean addPrice( nt 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( nt 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

EVOSUITE

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

UNIVERSITY OF LEICESTER

EVOSUITE

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

UNIVERSITY OF LEICESTER

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

EVOSUITE

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

UNIVERSITY OF LEICESTER

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

EVOSUITE

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

UNIVERSITY OF LEICESTER

EVOSUITE

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

UNIVERSITY OF LEICESTER

Coverage-Based Naming

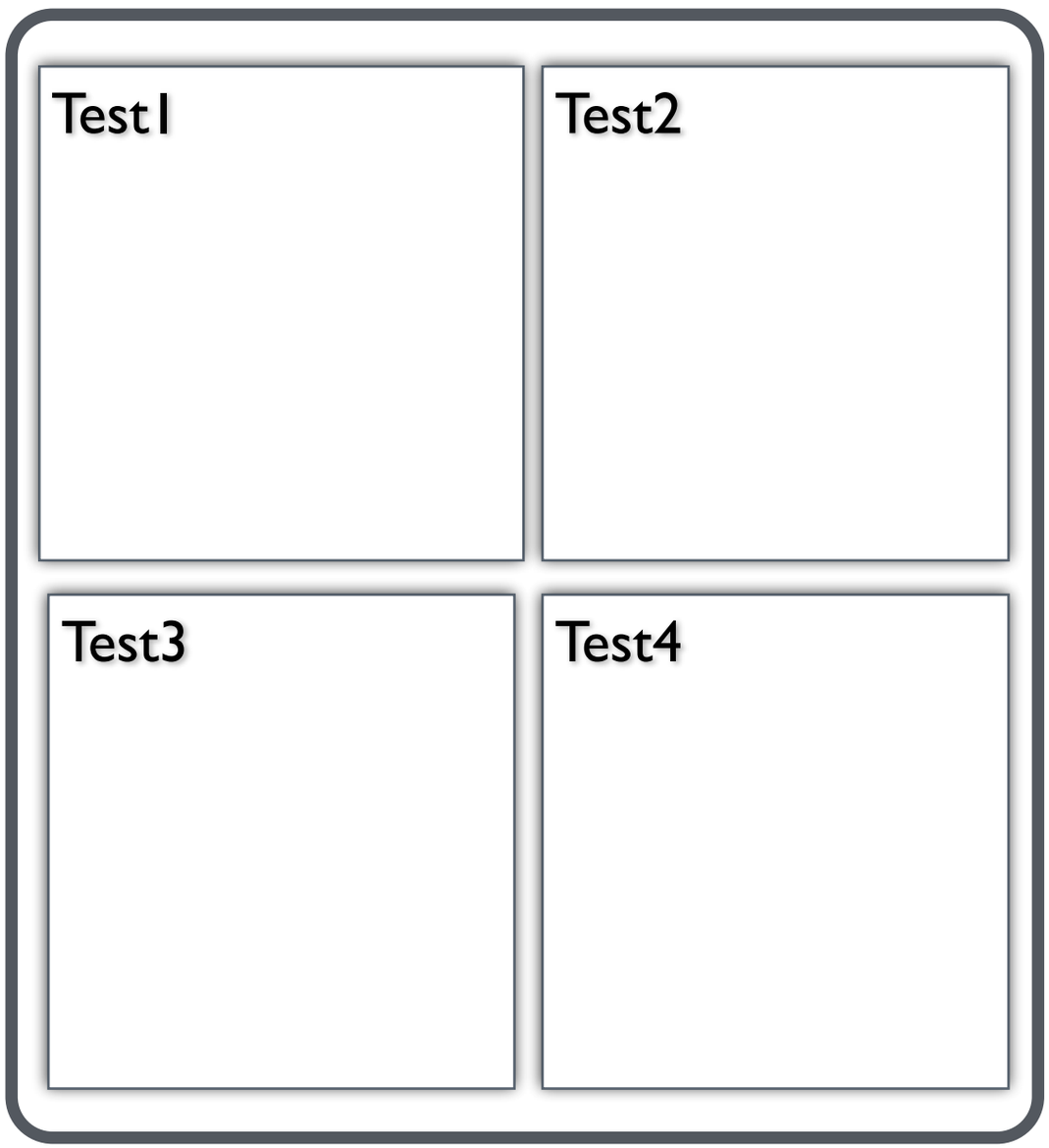
```
public class ShoppingCart {
   private int total = 0;
   private final static int MAX = 1000;

   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



Síntesis de Nombres de Casos de Prueba

Síntesis de Nombres de Casos de Prueba

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

Síntesis de Nombres de Casos de Prueba

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

Síntesis de Nombres de Casos de Prueba

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

Síntesis de Nombres de Casos de Prueba

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

Síntesis de Nombres de Casos de Prueba

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

Síntesis de Nombres de Casos de Prueba

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

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

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

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

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

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

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 entire pares de casts de prueba

Síntesis de Nombres de Casos de Prueba

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 entire pares de casts de prueba
5. Sufijos numéricos

EVOSUITE

DEMO

UNIVERSITY OF LEICESTER

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

UNIVERSITY OF LEICESTER

Evaluación Empírica

Estudio en línea

47 participantes

Evaluación Empírica

Estudio en línea

47 participantes

10
clases Java

UNIVERSITY OF LEICESTER

Evaluación Empírica

Estudio en línea

47 participantes

10
clases Java

2 expertos
como control

Evaluación Empírica

Estudio en línea

47 participantes

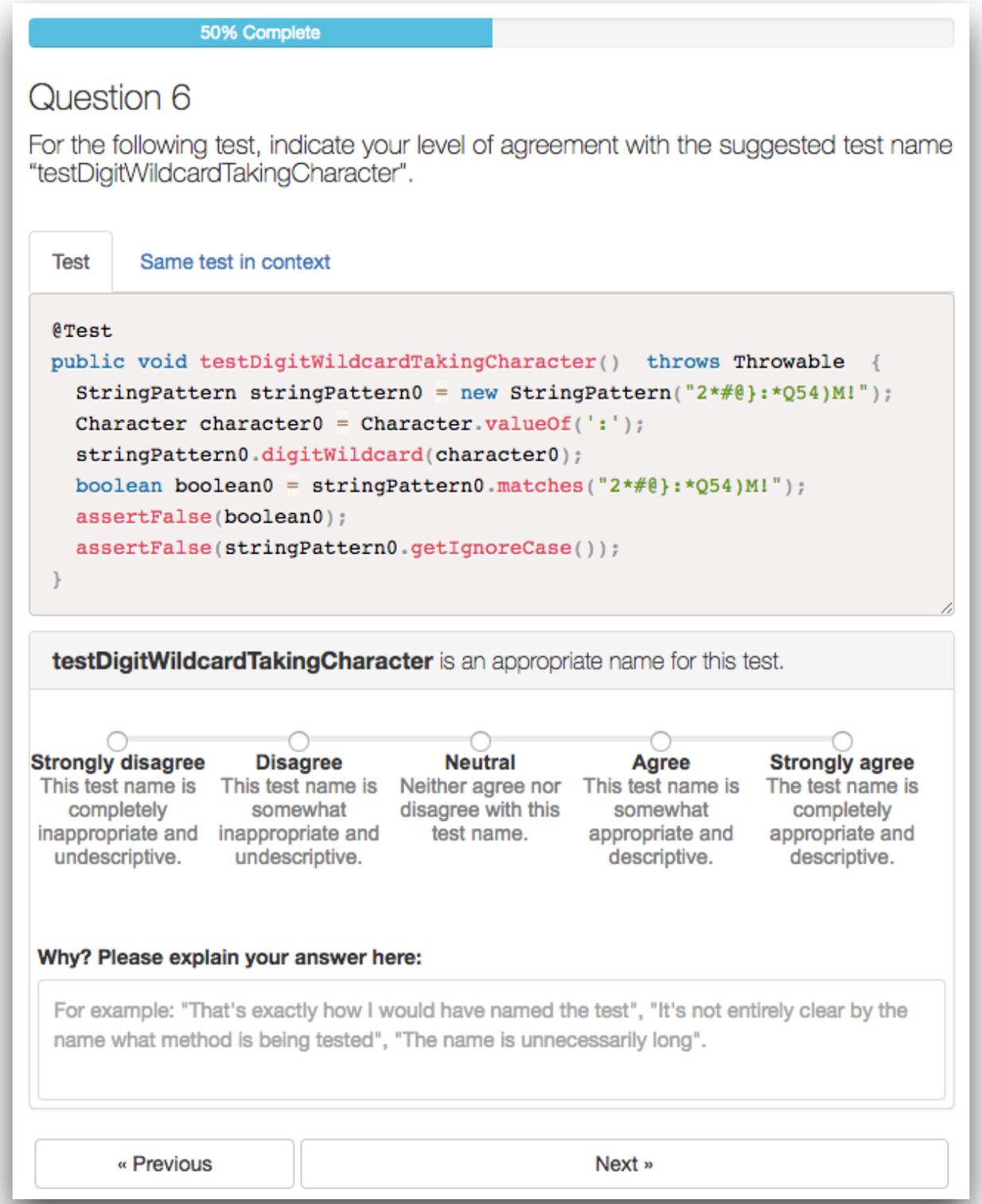
10
clases Java

2 expertos
como control

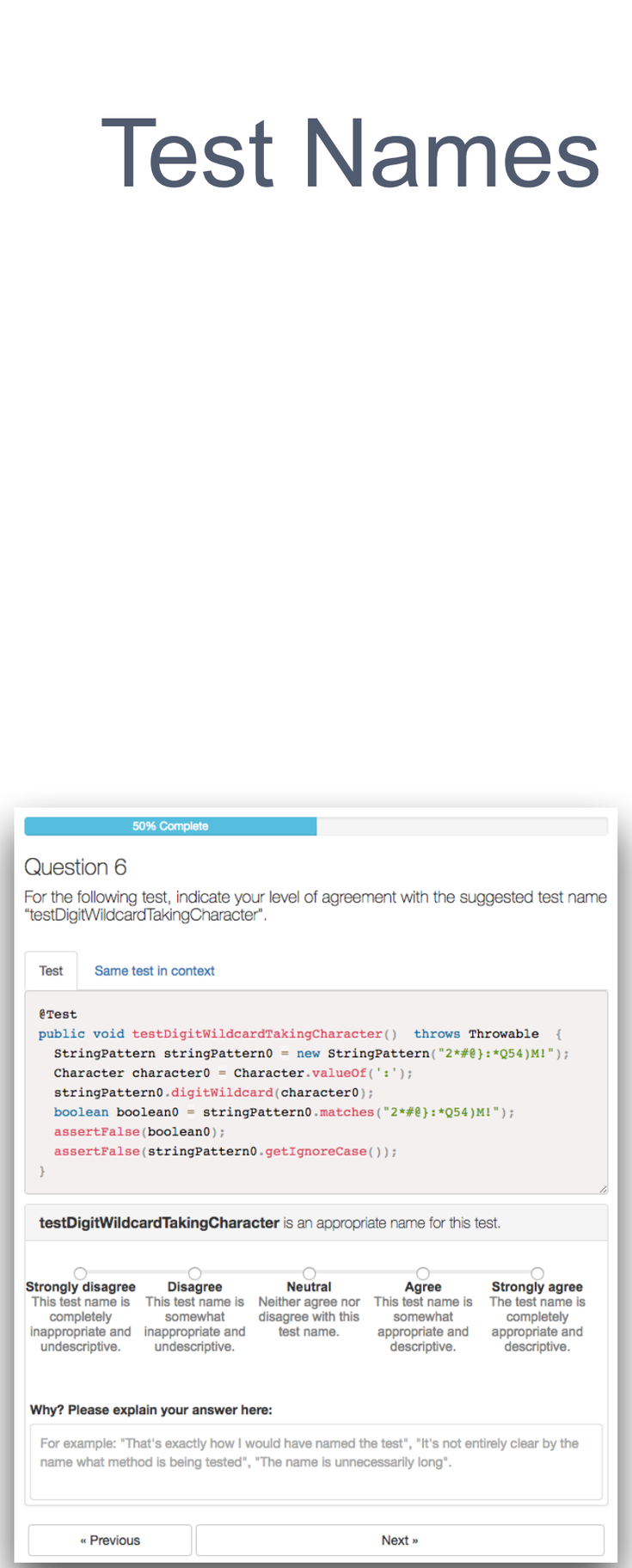
Sin tiempo límite

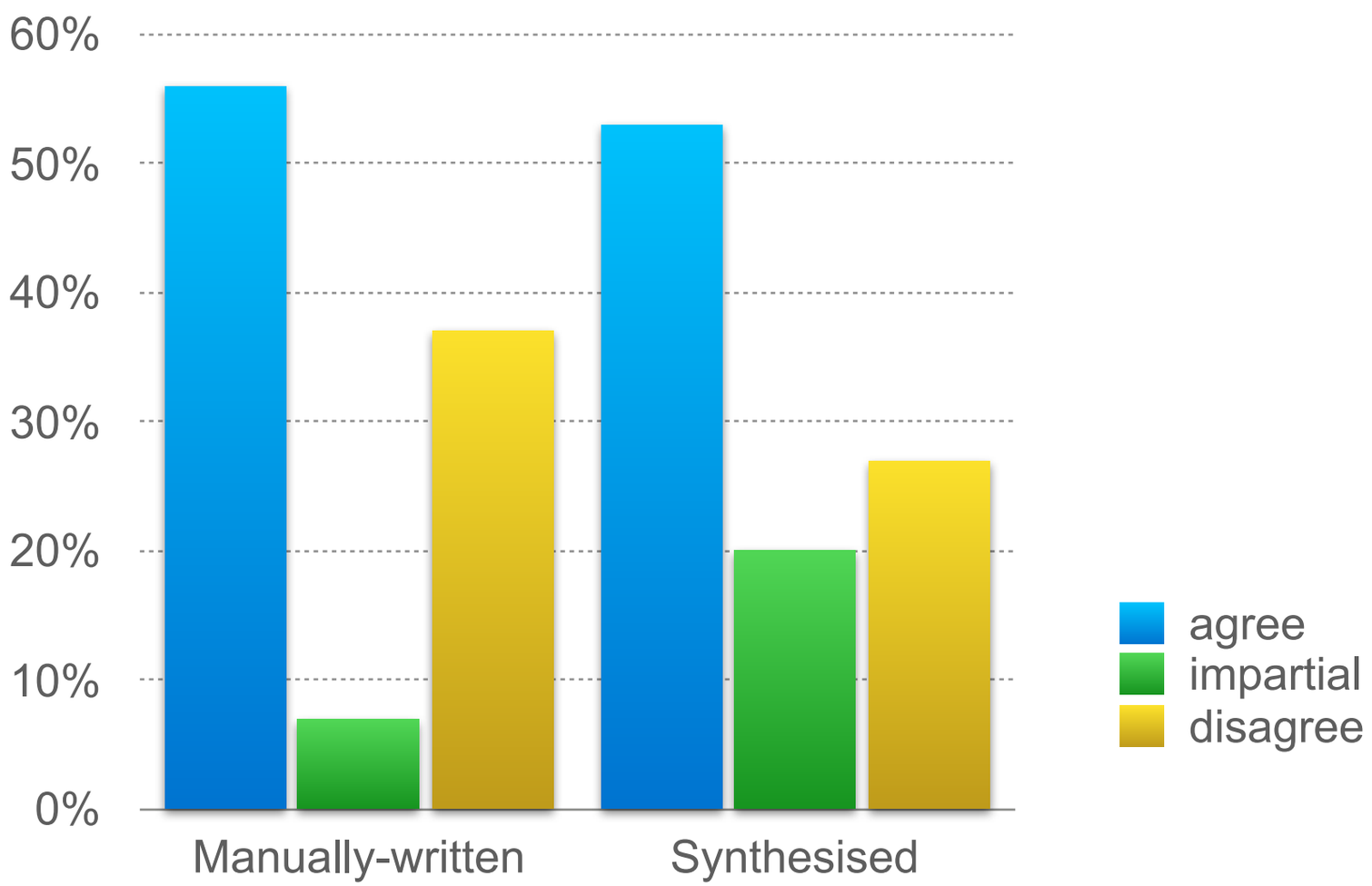
UNIVERSITY OF LEICESTER

Test Names



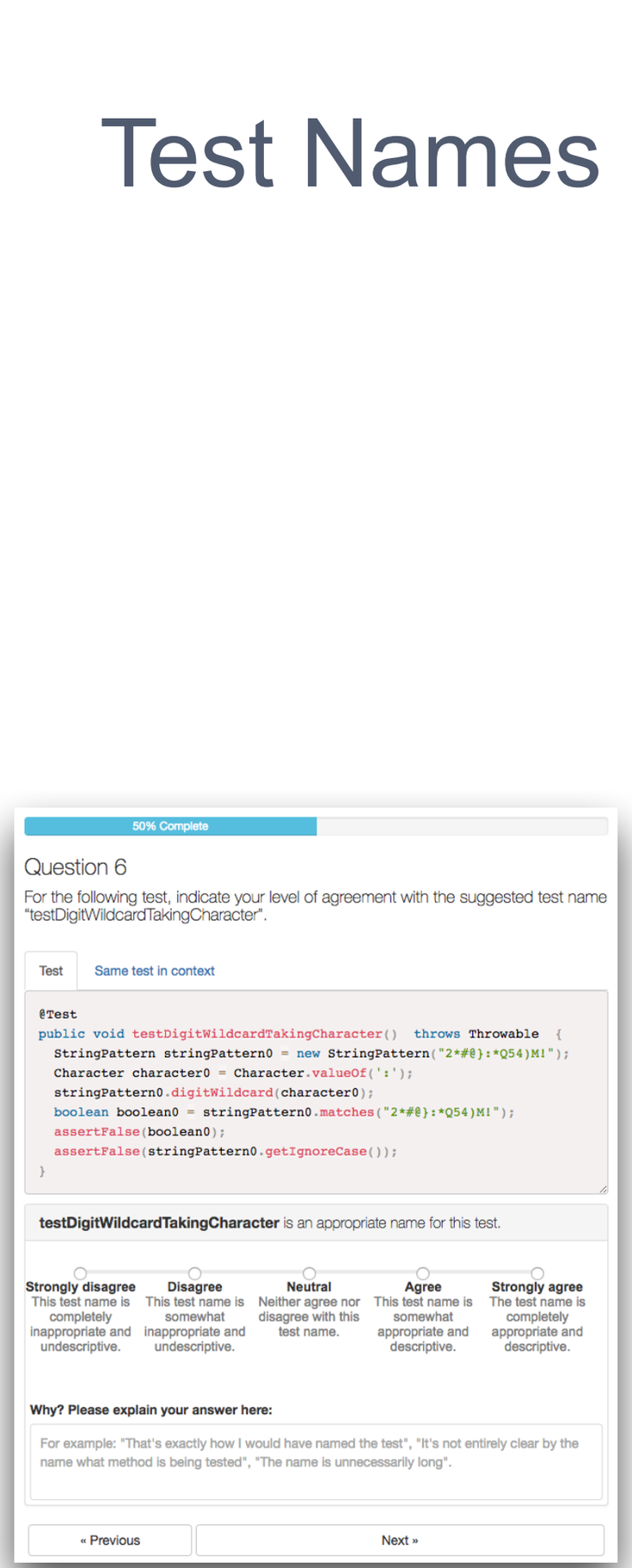
Test Names

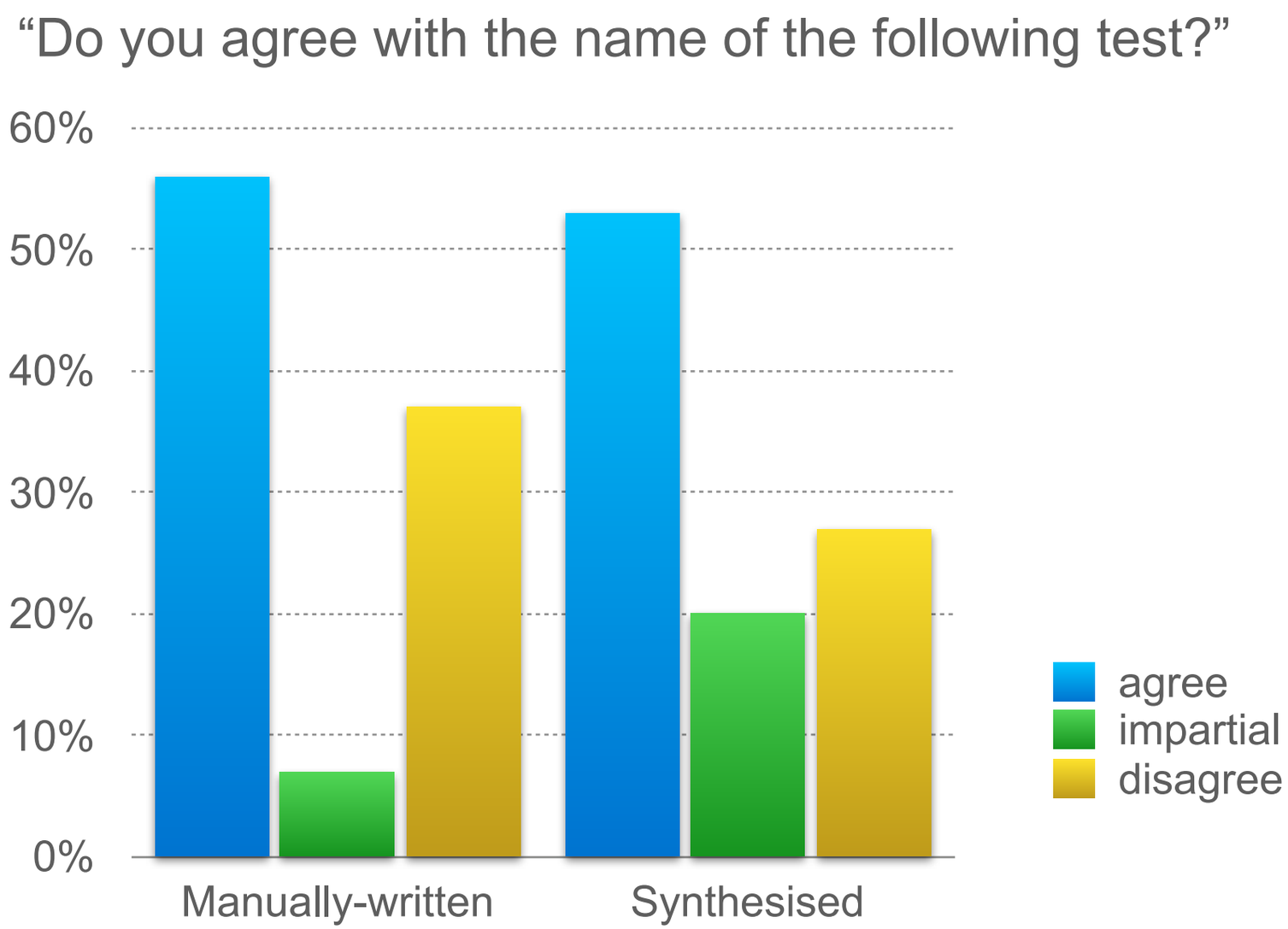




UNIVERSITY OF LEICESTER

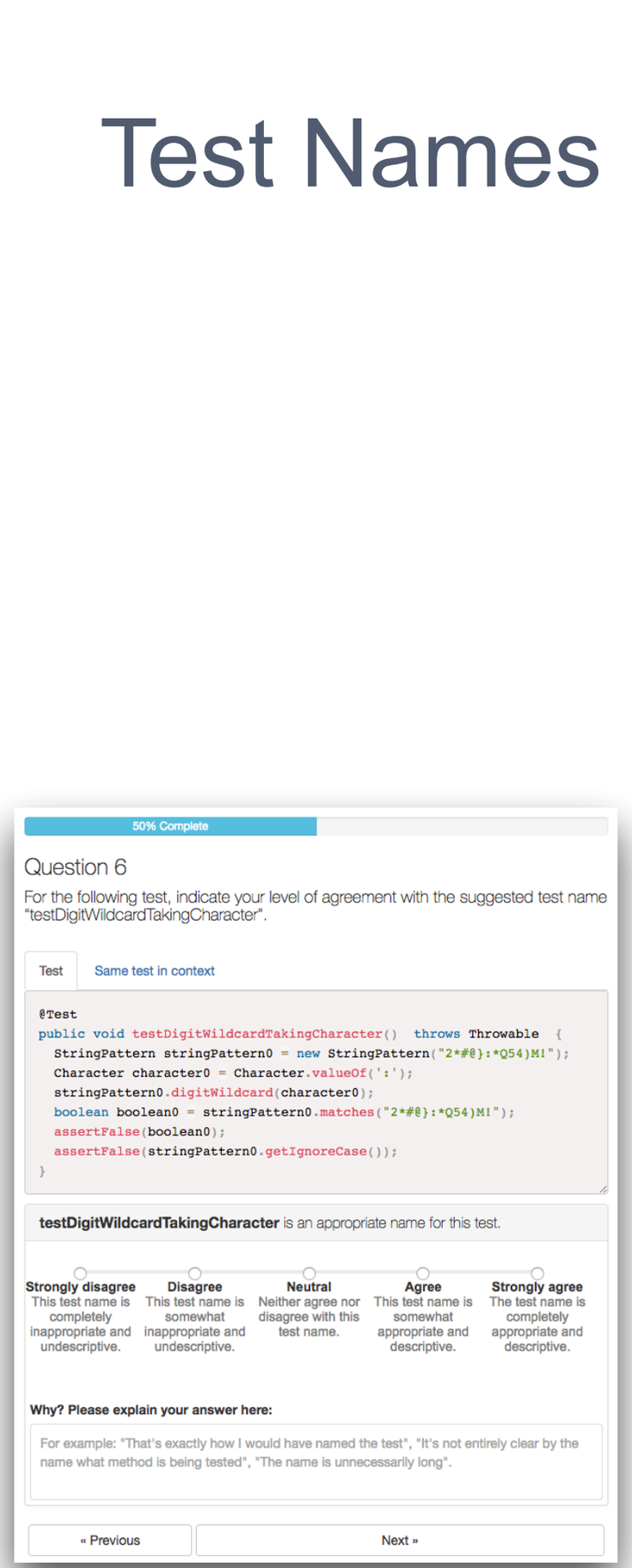
Test Names

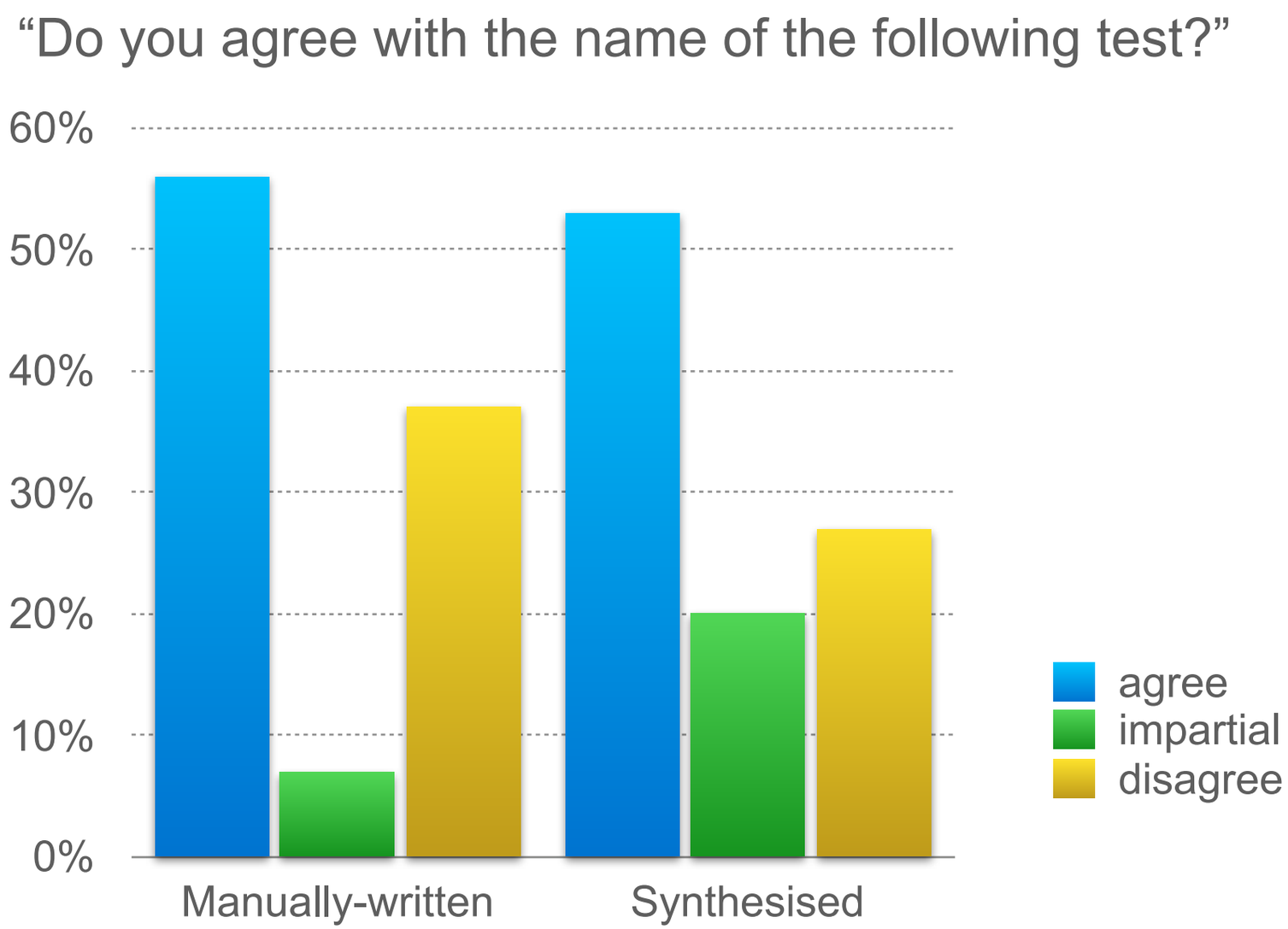




UNIVERSITY OF LEICESTER

Test Names





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

UNIVERSITY OF LEICESTER

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*#@}:*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.

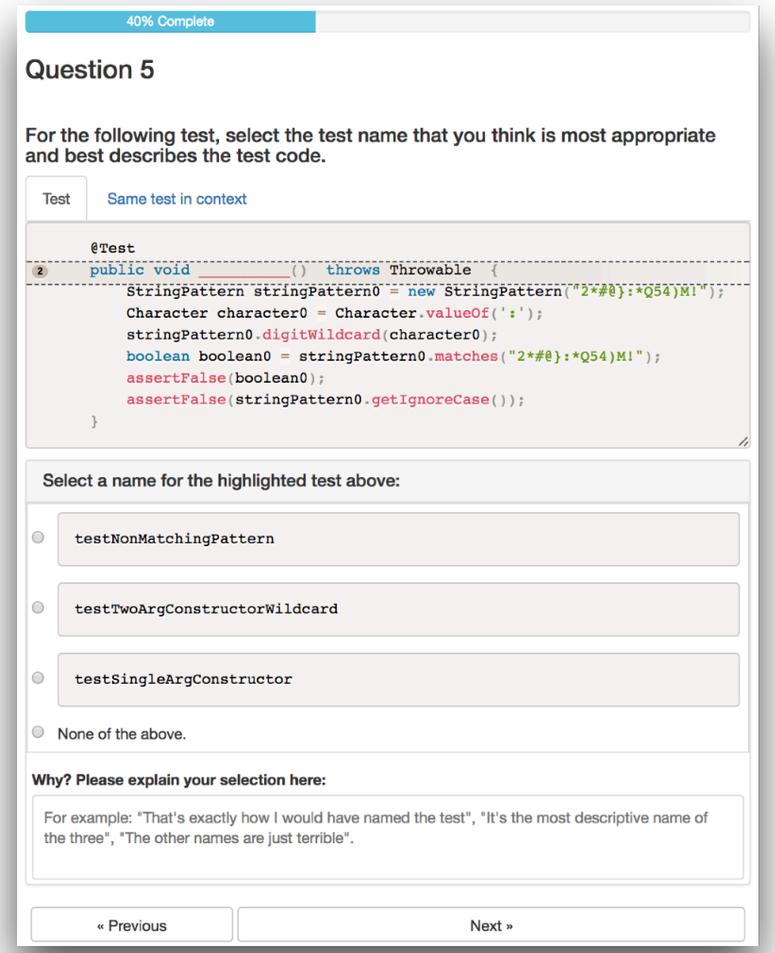
Why? Please explain your selection here:

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".

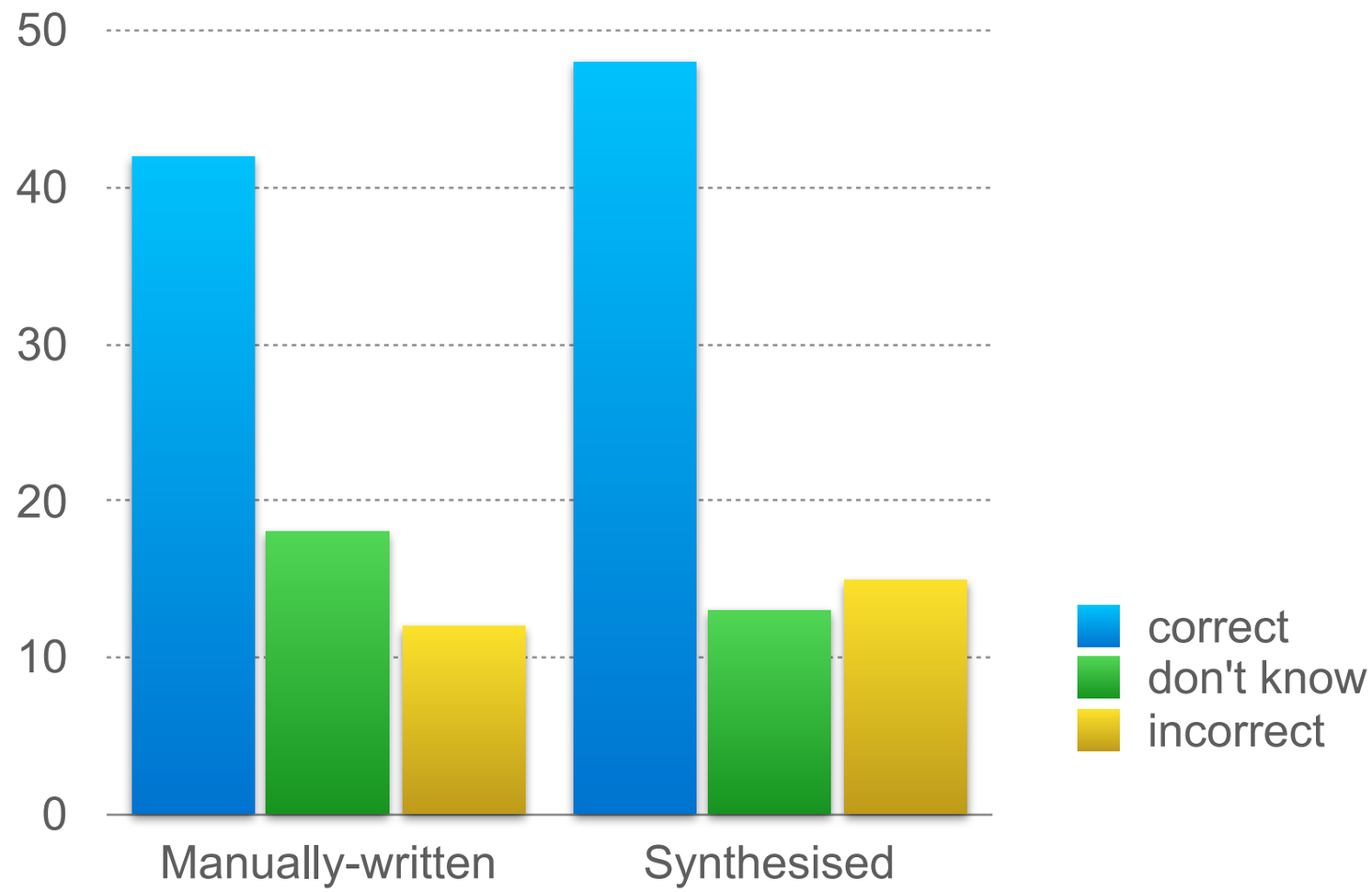
« Previous | Next »

UNIVERSITY OF LEICESTER

Test Names and Test Code

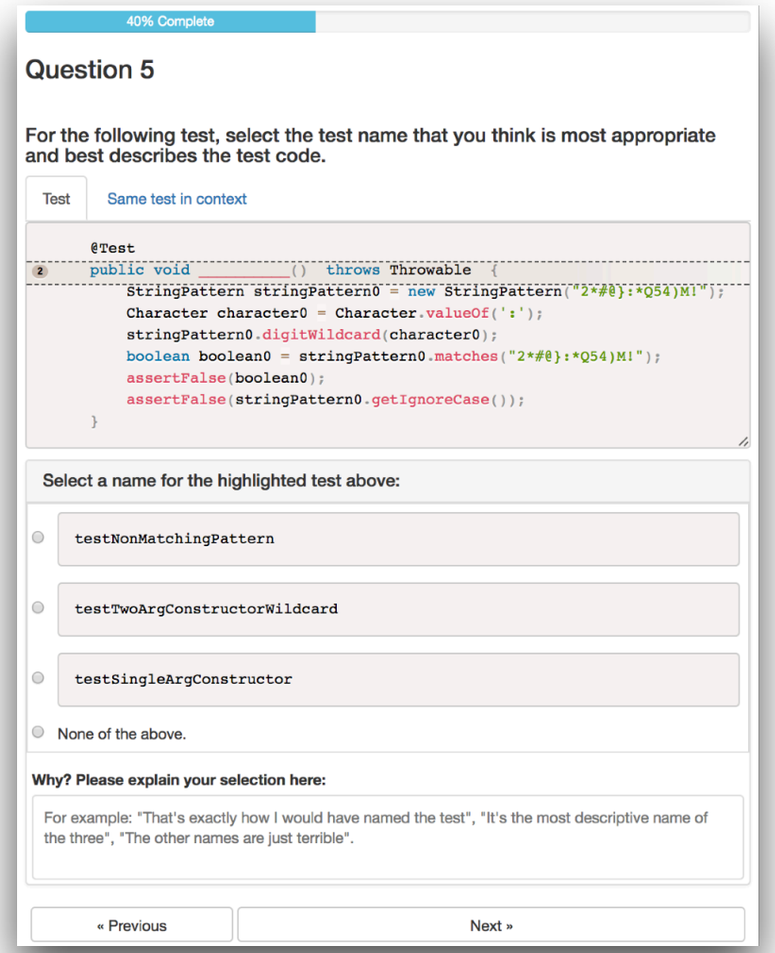


"For this test, select the most descriptive name from the list."

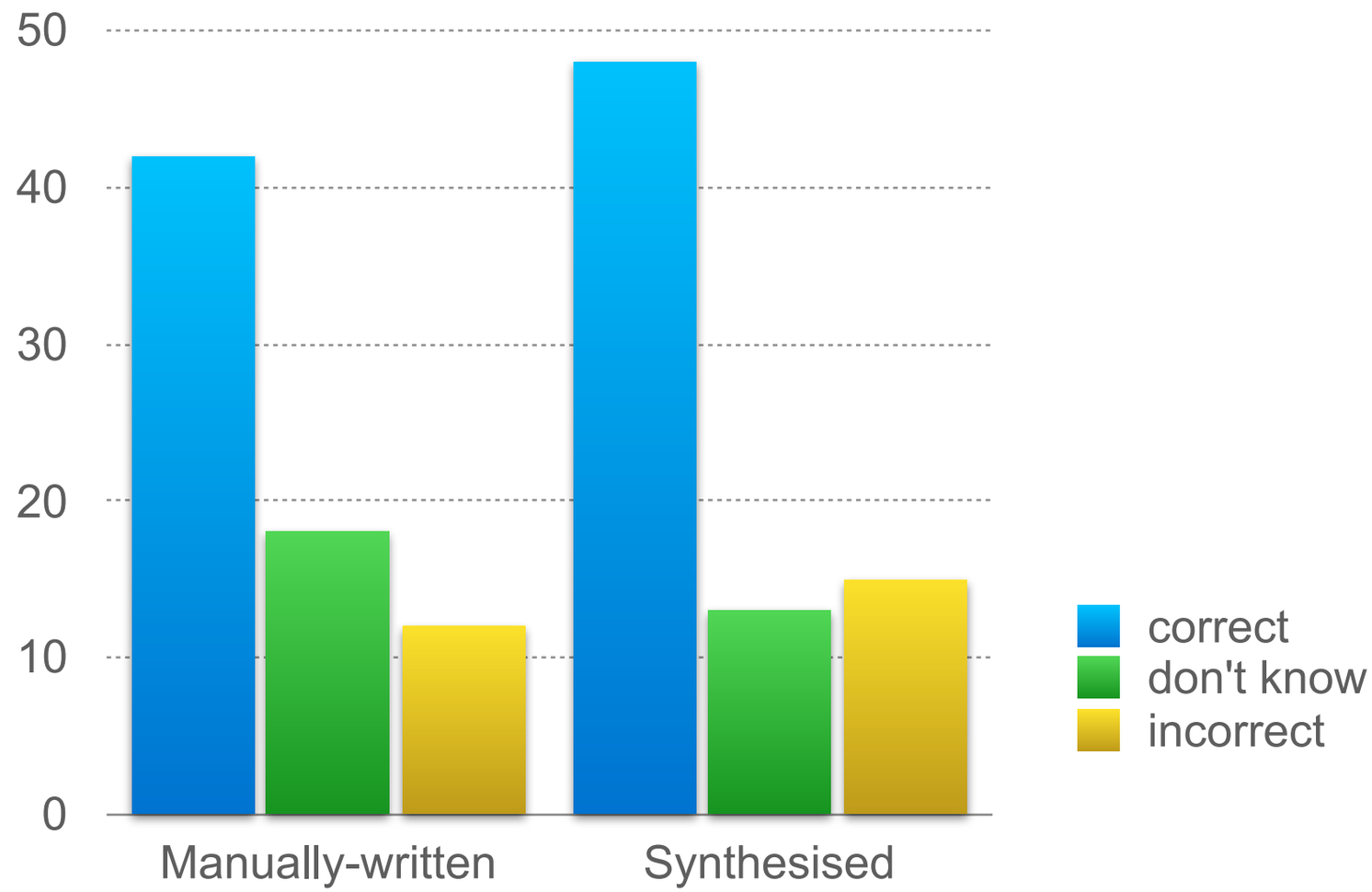


UNIVERSITY OF LEICESTER

Test Names and Test Code



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



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

UNIVERSITY OF LEICESTER

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:

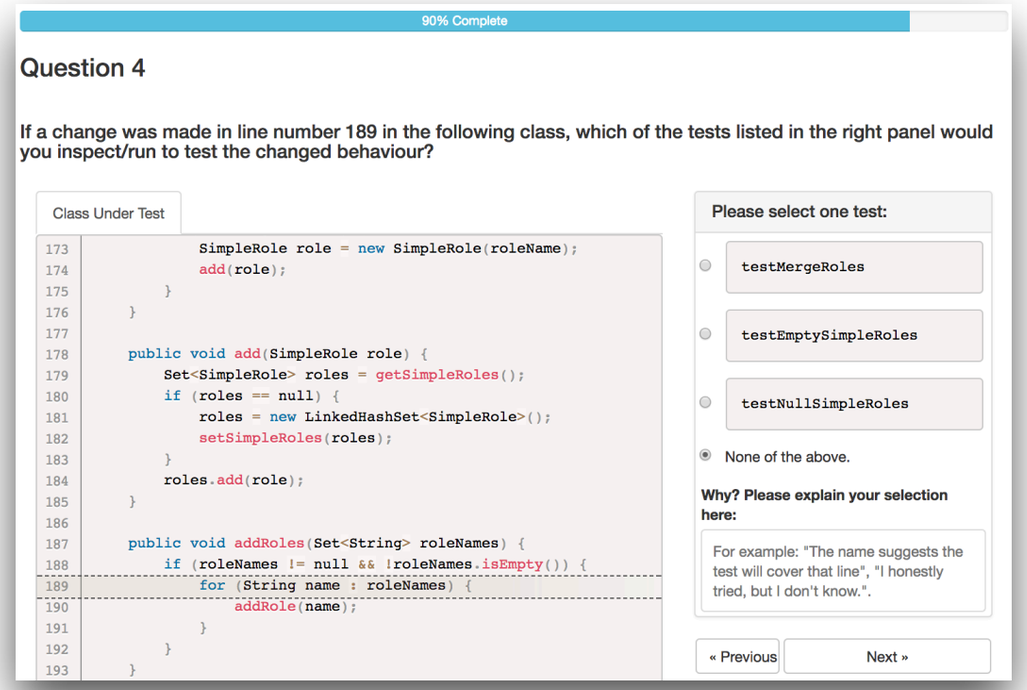
- `testMergeRoles`
- `testEmptySimpleRoles`
- `testNullSimpleRoles`
- (•) None of the above.

Why? Please explain your selection here:

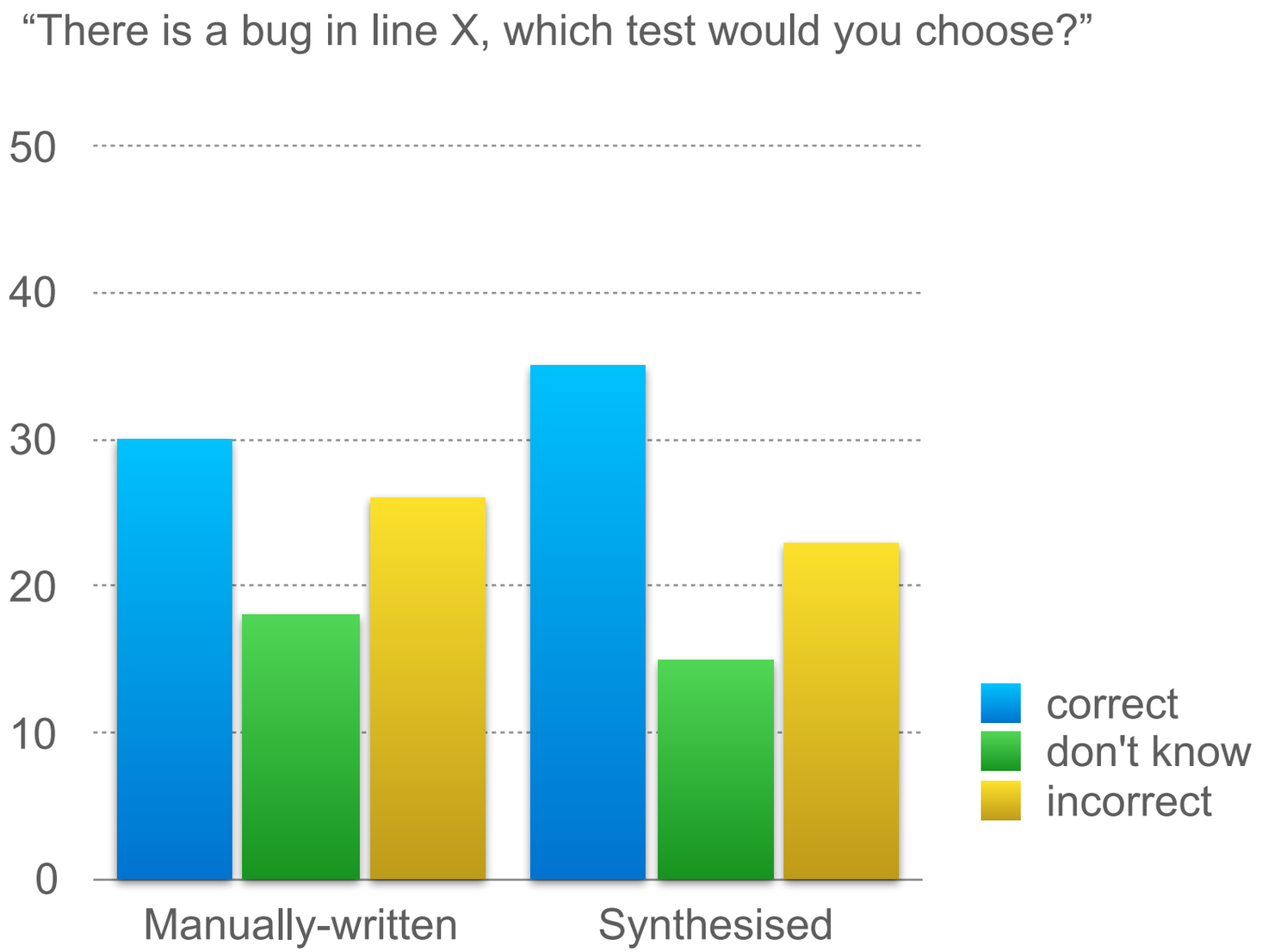
For example: "The name suggests the test will cover that line", "I honestly tried, but I don't know.".

« Previous | Next »

Test Names and Code under Test

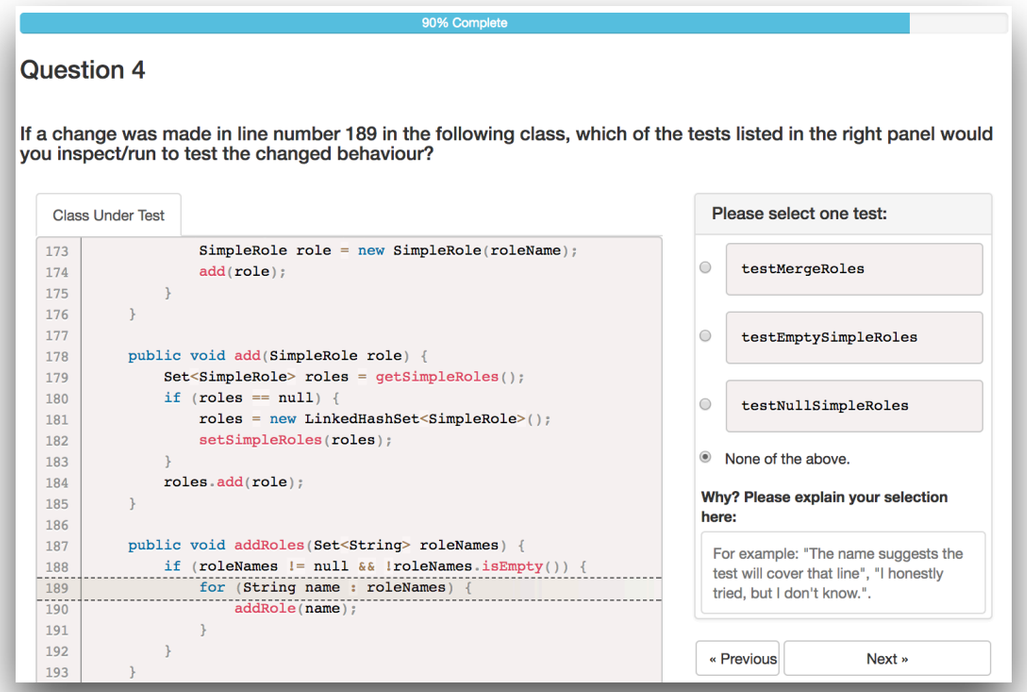


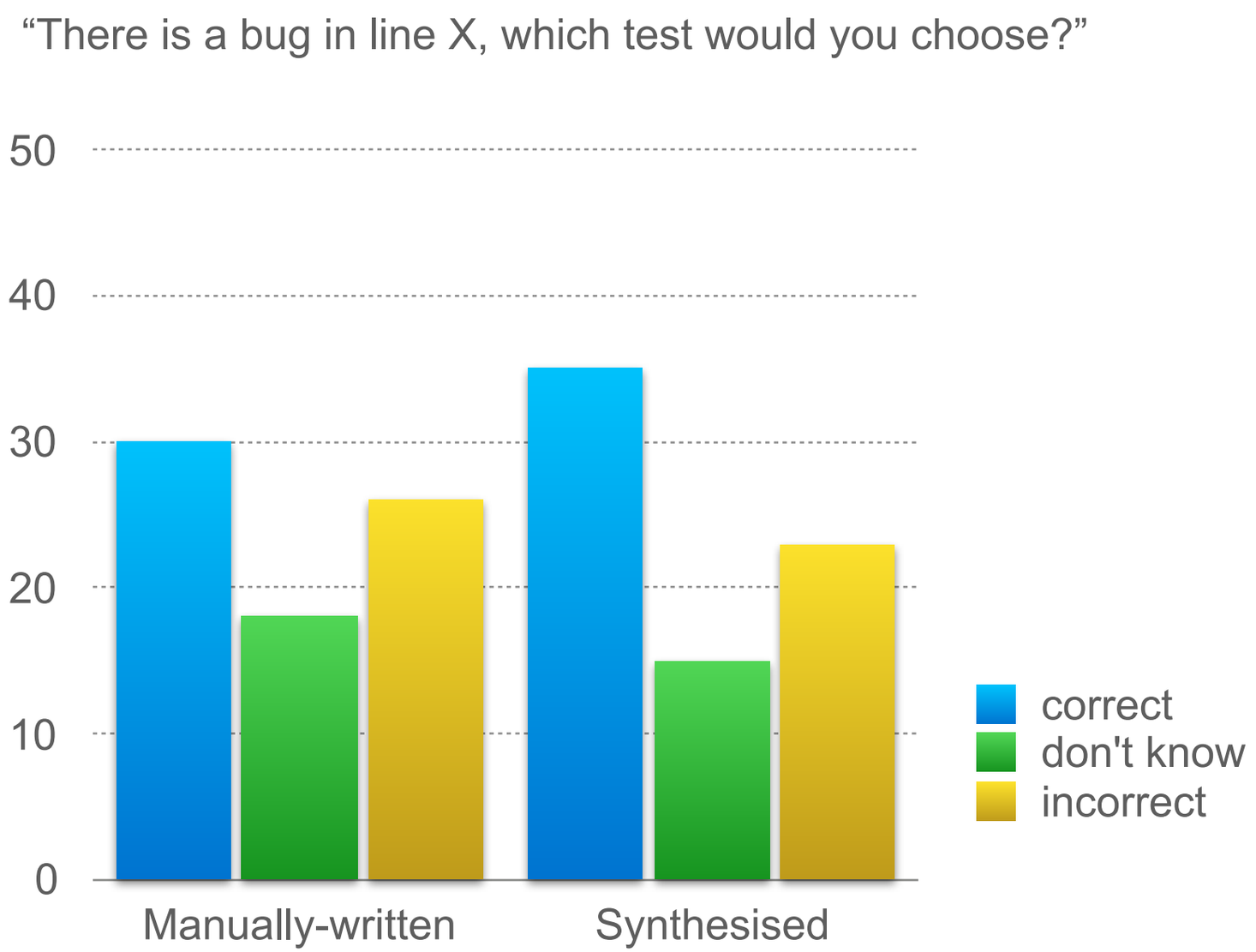
"There is a bug in line X, which test would you choose?"



UNIVERSITY OF LEICESTER

Test Names and Code under Test





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

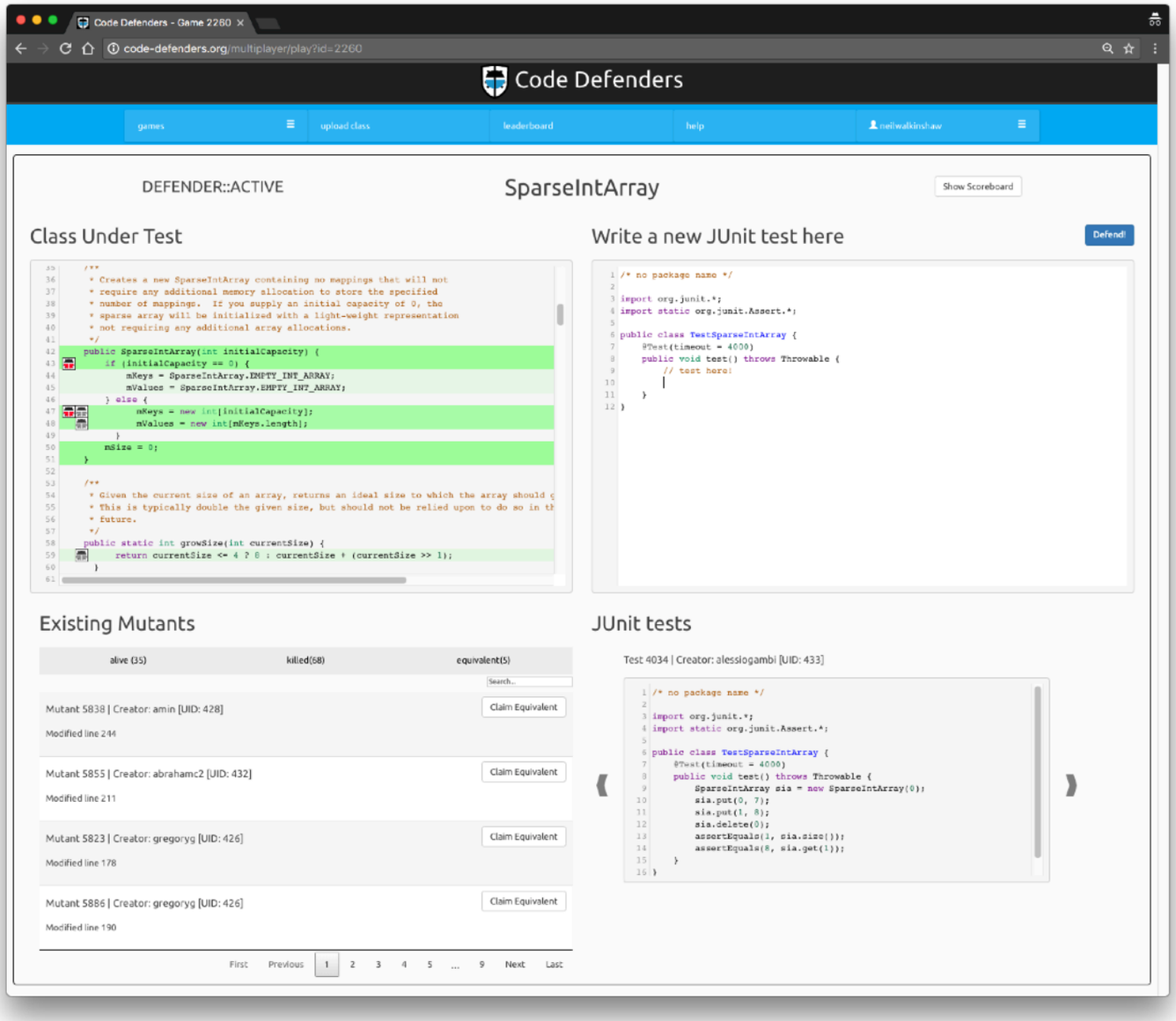
UNIVERSITY OF LEICESTER

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

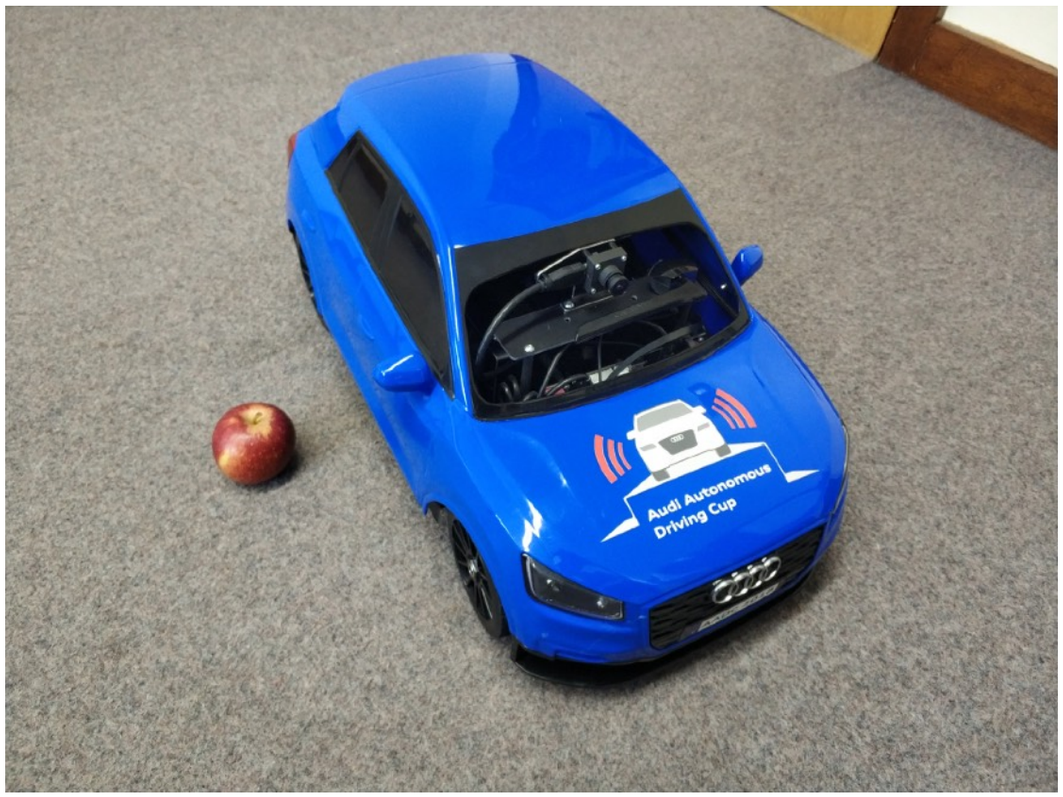
Software Engineering Gamification, e.g., Code Defenders

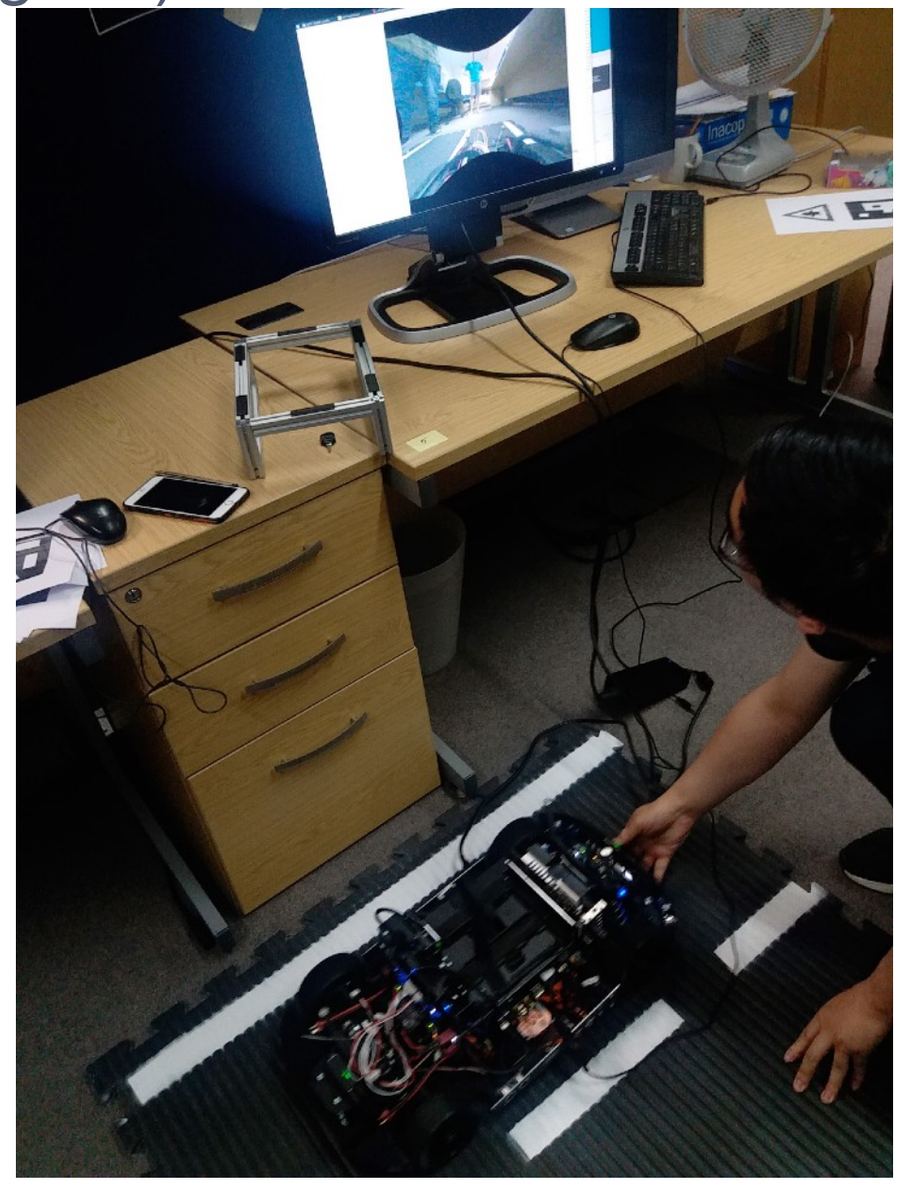


UNIVERSITY OF LEICESTER

Trabajo Relacionado

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



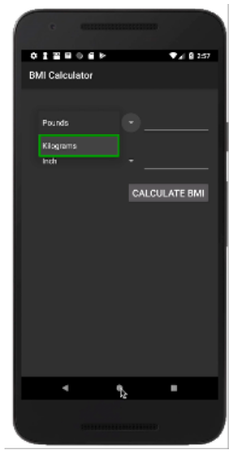


Trabajo Relacionado

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

Trabajo Relacionado

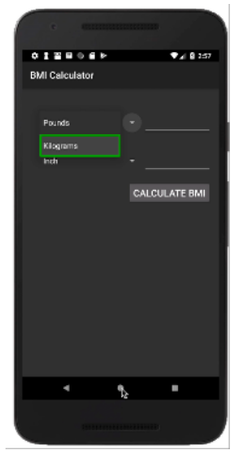
Pruebas no funcionales para apps móviles, e.g., Accessibilidad

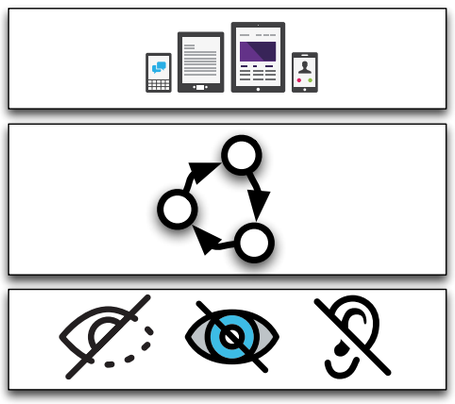


App under test

Trabajo Relacionado

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



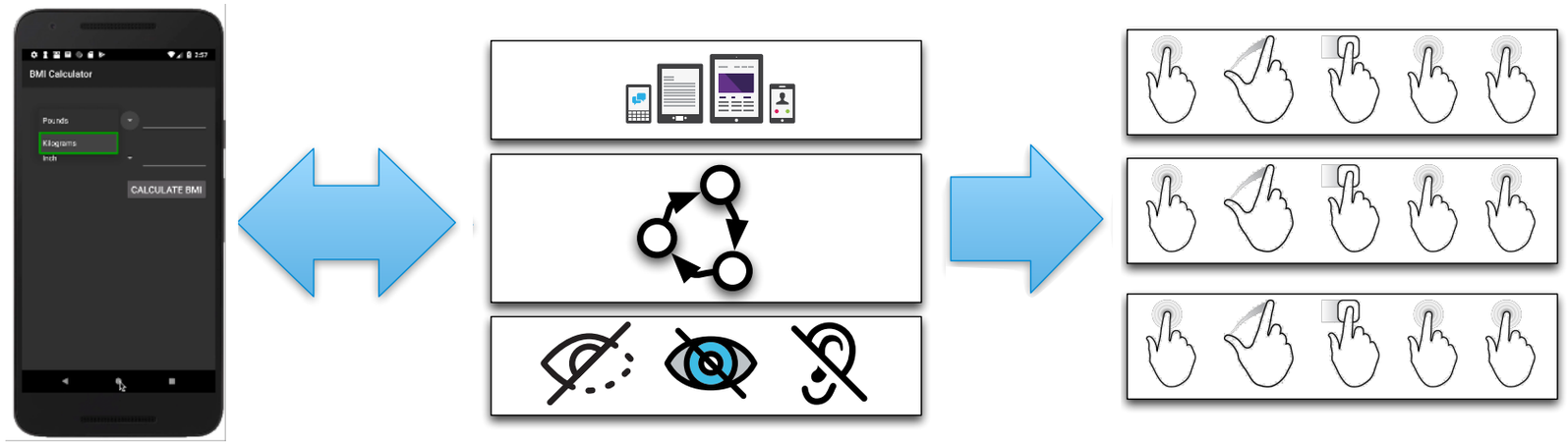


App under test

Test Generator

Trabajo Relacionado

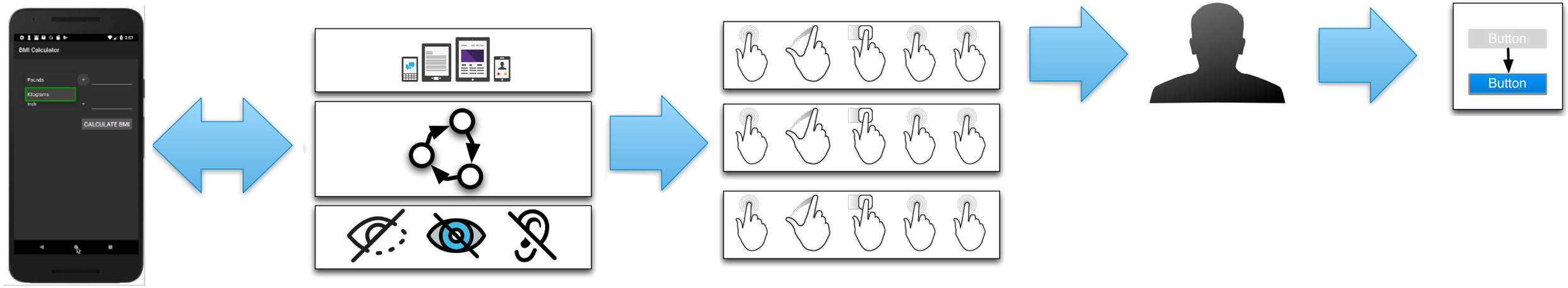
Pruebas no funcionales para apps móviles, e.g., Accessibilidad



App under test Test Generator Accessibility Tests

Trabajo Relacionado

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



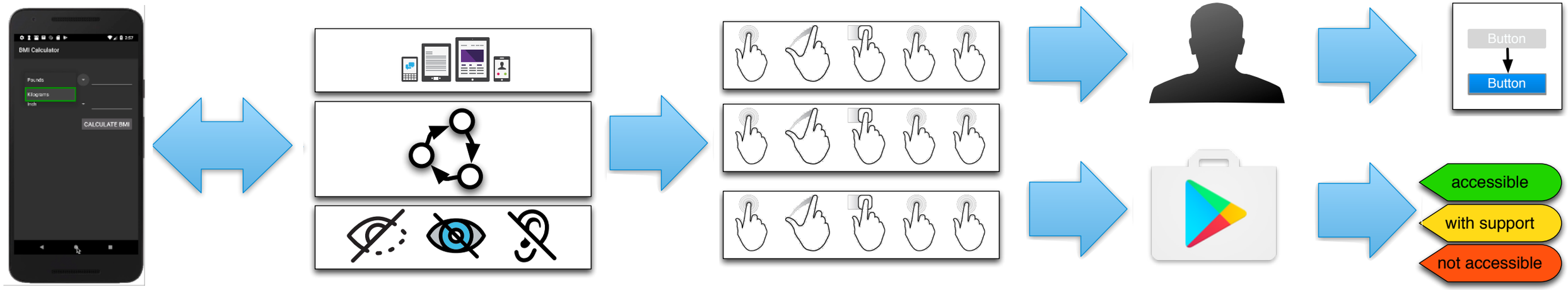
App under test

Test Generator

Accessibility Tests

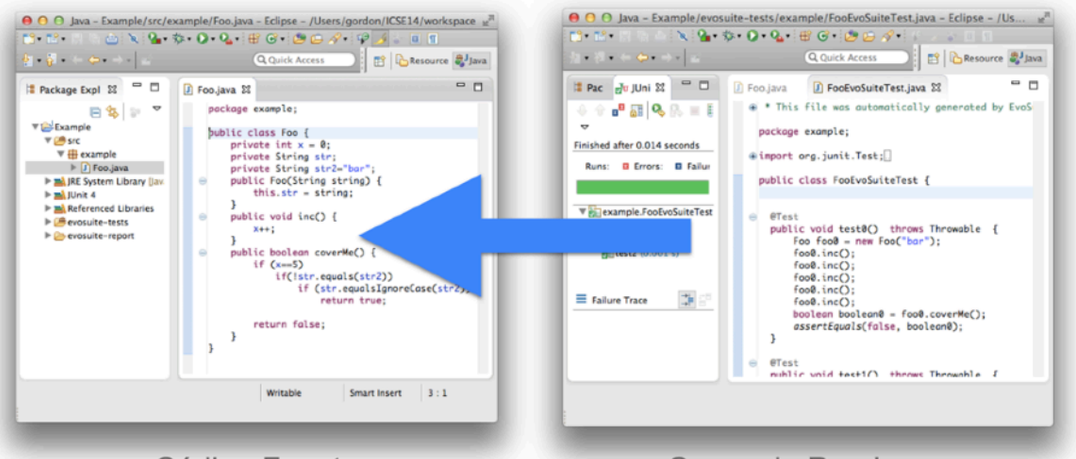
Trabajo Relacionado

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



App under test | Test Generator | Accessibility Tests | Developers/ App Store | Accessibility Fixes/Labels

EvoSuite



Código Fuente

Casos de Prueba

Generación Automática

UNIVERSITY OF LEICESTER

Evaluación Empírica

Estudio en línea

47 participantes

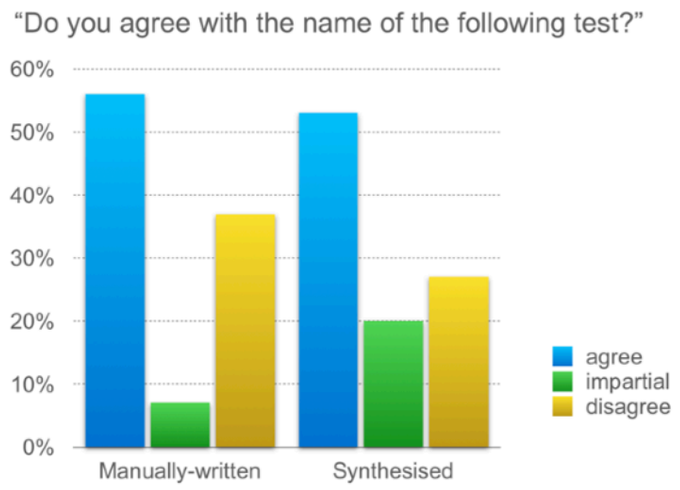
10
clases Java

2 expertos
como control

Sin tiempo límite

UNIVERSITY OF LEICESTER

Nombres de Tests



UNIVERSITY OF LEICESTER

evosuite.org

j.rojas@leicester.ac.uk