Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra de Ingeniería de Software

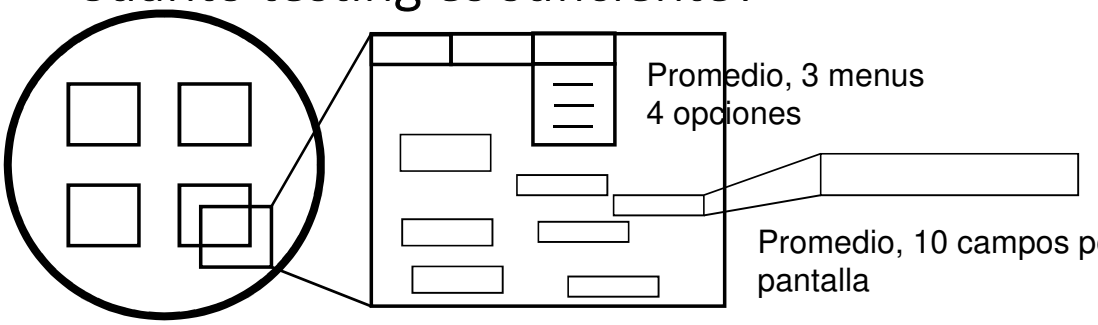
TESTING DE SOFTWARE

Consuelo López

Qué es el testing?

2

Cuánto testing es suficiente?



Sistema con 20 pantallas

Numero de dos dígitos, 101 valores posibles

Total de testing exhaustivo:
- 20 x 3 x 4 x 10 x 100 = 240.000

Suponiendo 1 seg por prueba:
4000 minutos -> 67 horas -> 8,5 días

10 seg -> 17 semanas 1 min -> 1,4 años 10 min -> 13,7 años

3

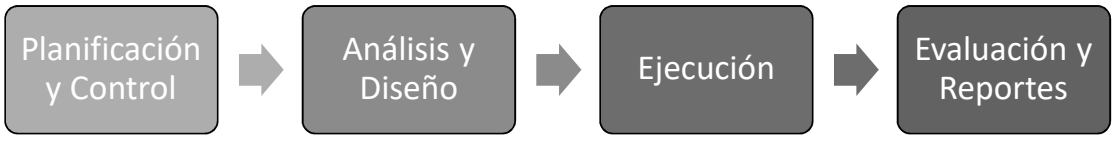
La Psicología del testing: Desarrolladores vs Testers

4

Conceptos: Error vs Defecto

5

Proceso del Testing

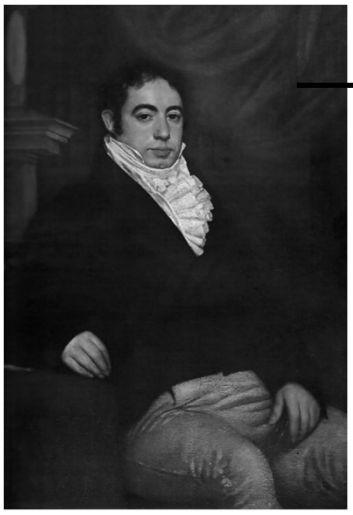


6

Niveles de Testing

7

Niveles de Testing: Testing Unitario



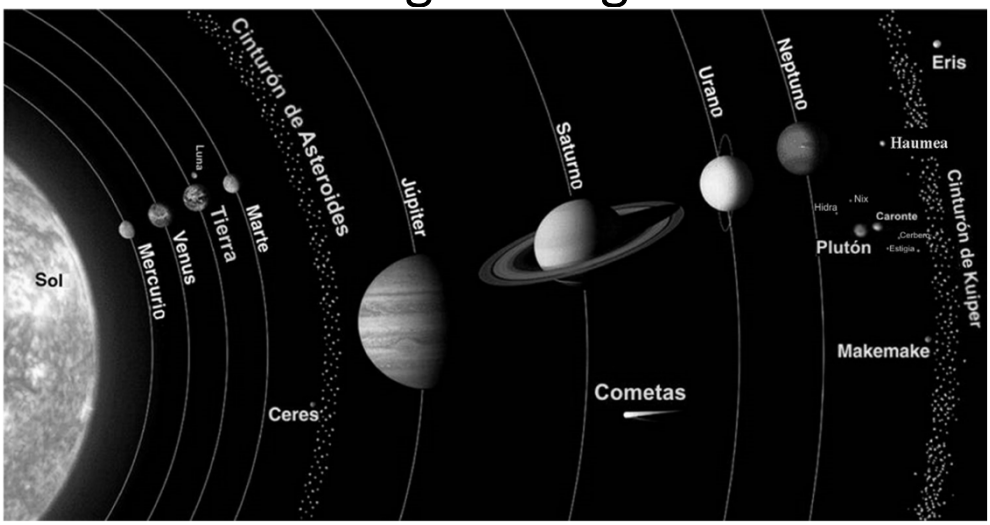
Bernardino Rivadavia

8

Niveles de Testing: Testing de Integración

9

Niveles de Testing: Testing de Sistema

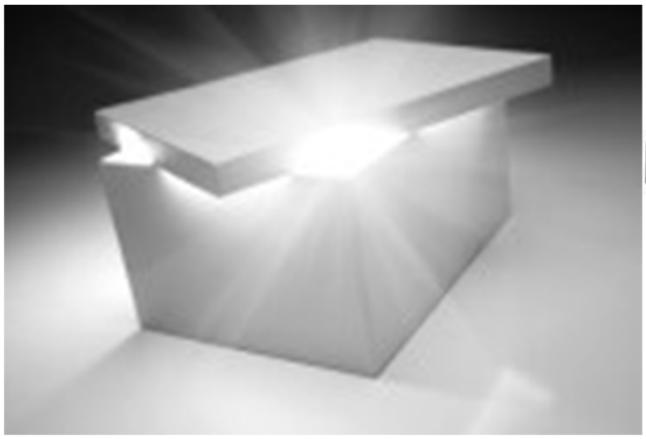


10

Niveles de Testing: Testing de Aceptación

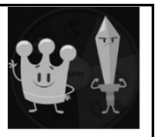
11

Estrategias

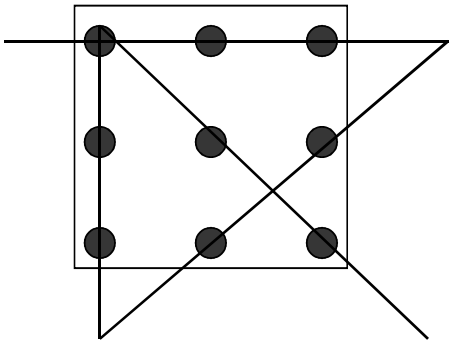


12

Métodos: Quick Quiz



- Unir todos los puntos usando solamente 4 trazos



Think out of the box!!

13

Métodos

- Para qué usarlos? El tiempo y el presupuesto es limitado
- Hay que pasar por la mayor cantidad de funcionalidades con la menor cantidad de pruebas

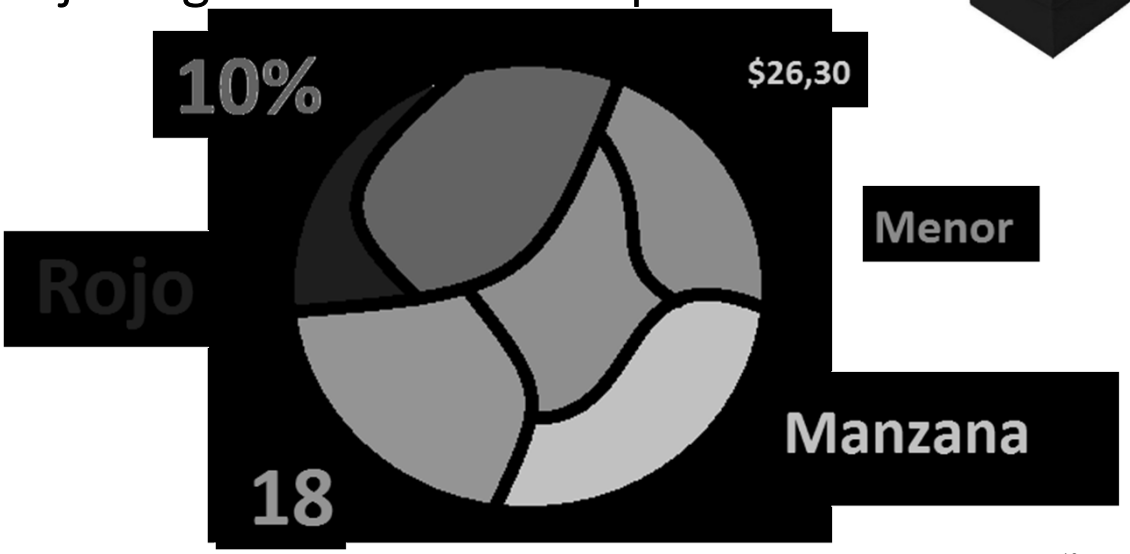
14

Caja Negra

- Basado en especificaciones
  - Partición de Equivalencias
  - Análisis de valores límites
  - Etc.
- Basados en la experiencia
  - Adivinanza de Defectos
  - Testing Exploratorio

15

Caja Negra: Partición de Equivalencias

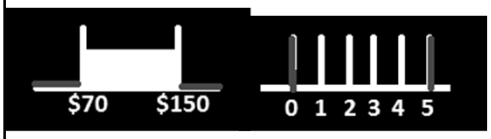


16

Caja Negra: Partición de Equivalencias

Dos Pasos

1. **Identificar las clases de equivalencia (Válidas y no Válidas)**
   - Rango de valores continuos
   - Valores Discretos
   - Selección simple
   - Selección múltiple



2. **Identificar los casos de prueba**

17

Caja Negra: Partición de Equivalencias

1. **Un empleado puede percibir hasta $4000 sin pagar impuestos**
2. **Para los siguientes $1500, el impuesto es del 10% del total.**
3. **Para los próximos $2000, el impuesto aplicado es del 22%**
4. **Cualquier monto superior percibirá un 40% de deducciones sobre el total.**

18

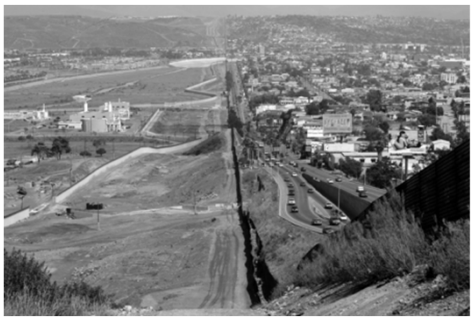
Caja Negra: Partición de Equivalencias

1. **Solución!!!**

19

Caja Negra: Análisis de Valores límites

- Es una variante de la partición de equivalencias, en vez de seleccionar cualquier elemento como representativo de una clase de equivalencia, se seleccionan los bordes de una clase.



20

Caja Negra: Análisis de Valores límites

- Plantear los Casos de Prueba anteriormente descriptos para el método de Análisis de Valores Límites.

21

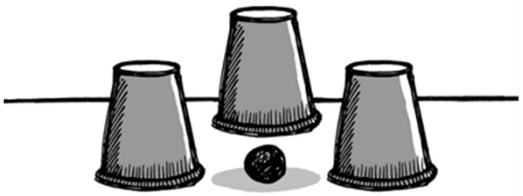
Caja Negra: Análisis de Valores límites

- Solución!

22

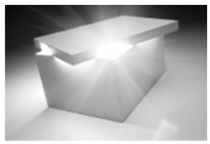
Caja Negra: Basados en la experiencia

- Adivinanza de Defectos
- Testing Exploratorio



23

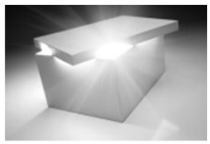
Caja Blanca



- Se basan en el análisis de la estructura interna del software o un componente del software.
- Se puede garantizar el testing coverage

24

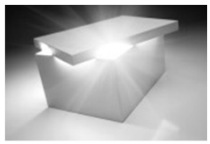
Caja Blanca



- Cobertura de enunciados o caminos básicos
- Cobertura de sentencias
- Cobertura de decisión
- Cobertura de condición
- Cobertura de decisión/condición
- Cobertura múltiple
- Etc

25

Caja Blanca



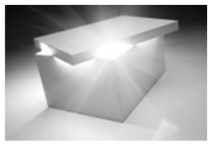
Cobertura de enunciados o caminos básicos

- Propuesto por McCabe
- Permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución

Para la prueba del camino básico:

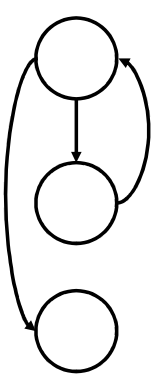
- Se requiere poder representar la ejecución mediante grafos de flujo
- Se calcula la complejidad ciclomática
- Dado un grafo de flujo se pueden generar casos de prueba

Caja Blanca

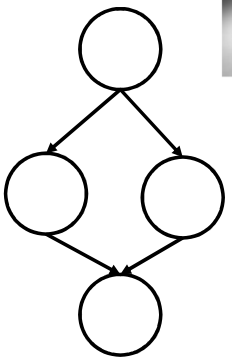


Cobertura de enunciados o caminos básicos

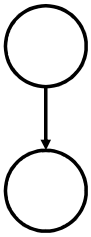
Grafo de flujos de Estructuras básicas



While

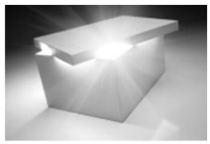


If…Else…



Secuencia

Caja Blanca



Cobertura de enunciados o caminos básicos

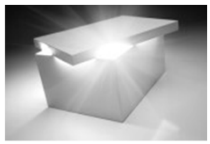
Complejidad Ciclomática

- Es una métrica de software que provee una medición cuantitativa de la complejidad lógica de un programa
- Usada en el contexto de testing, define el número de caminos independientes en el conjunto básico y entrega un limite inferior para el número de casos necesarios para ejecutar todas las instrucciones al menos una vez

M = Complejidad ciclomática.
E = Número de aristas del grafo
N = Número de nodos del grafo
P = Número de componentes conexos, nodos de salida

$M = E - N + 2*P$

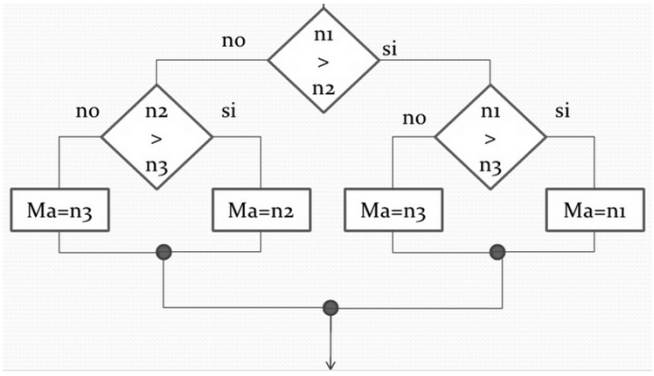
M = Número de regiones + 1

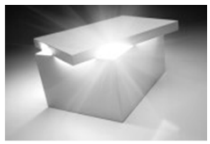


Caja Blanca

Cobertura de enunciados o caminos básicos

Ejemplo





Caja Blanca

Cobertura de enunciados o caminos básicos

M = E – N + 2*P

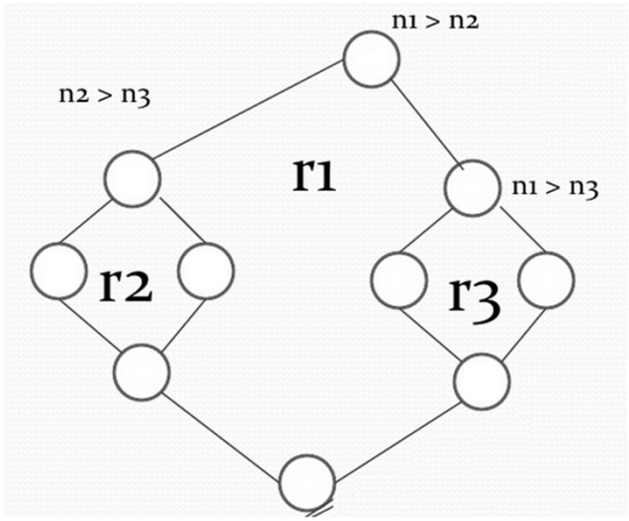
M = 12 – 10 + 2*1

M = 4

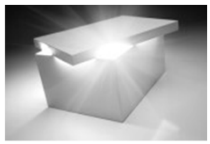
M = Número de regiones + 1

M = 3 + 1

M = 4



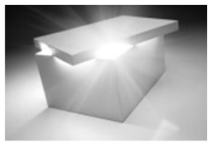
Caja Blanca



Cobertura de enunciados o caminos básicos

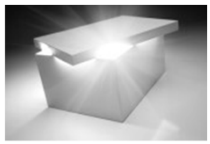
TC 1	TC 2	TC 3	TC 4
N1 = 8	N1 = 8	N1 = 4	N1 = 4
N2 = 4	N2 = 4	N2 = 8	N2 = 8
N3 = 4	N3 = 8	N3 = 4	N3 = 8

Caja Blanca



Cobertura de enunciados o caminos básicos

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

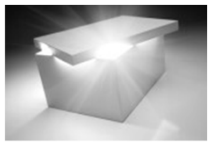


Caja Blanca

Cobertura de enunciados o caminos básicos

Pasos del diseño de pruebas mediante el camino básico

- Obtener el grafo de flujo
- Obtener la complejidad ciclomática del grafo de flujo
- Definir el conjunto básico de caminos independientes
- Determinar los casos de prueba que permitan la ejecución de cada uno de los caminos anteriores
- Ejecutar cada caso de prueba y comprobar que los resultados son los esperado



Caja Blanca

Cobertura de sentencias

```
IF (A>0 && C==1)
        X = X + 1
IF (B==3 || D<0)
        Y=0
END
```

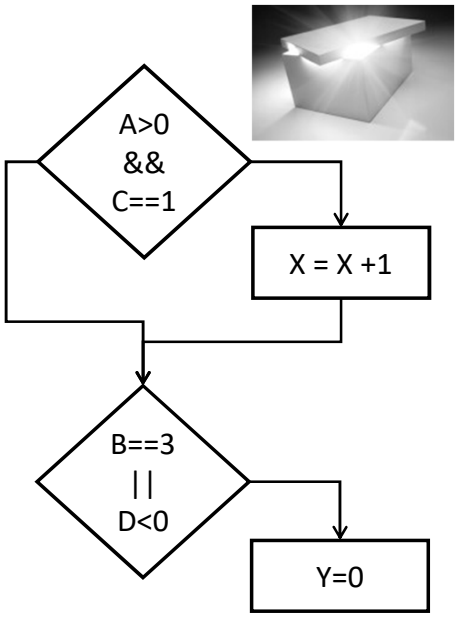
TC1
A=5
C=1
B=3
D=-3

Caja Blanca

Cobertura de decisión

```
IF (A>0 && C==1)
        X = X + 1
IF (B==3 || D<0)
        Y=0
END
```

TC1	TC2
A=5 C=1 B=3 D=-3	A=5 C=5 B=5 D=5

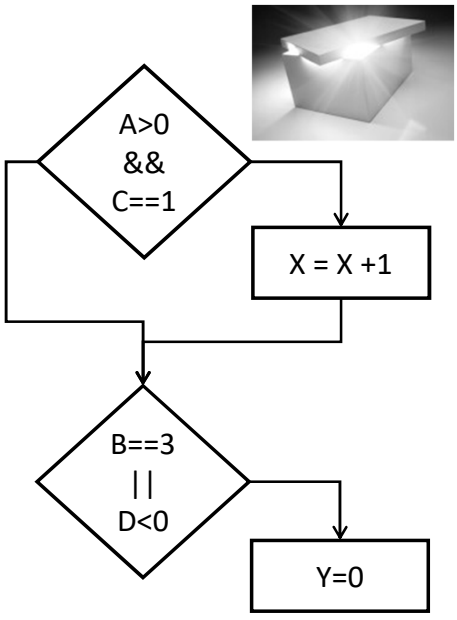


Caja Blanca

Cobertura de condición

```
IF (A>0 && C==1)
        X = X + 1
IF (B==3 || D<0)
        Y=0
END
```

TC1	TC2
A=0 C=1 B=3 D=-3	A=5 C=5 B=5 D=5

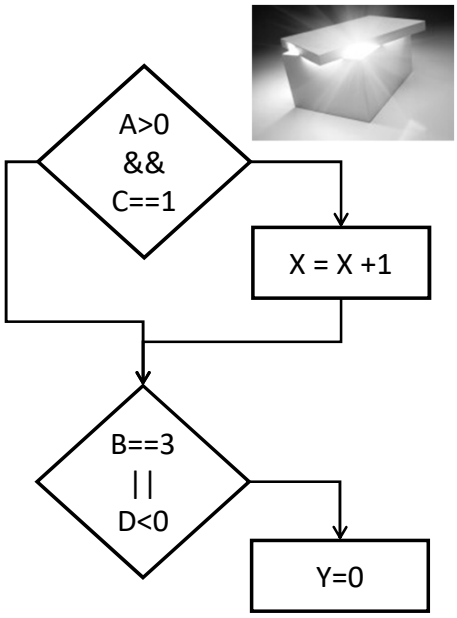


Caja Blanca

Cobertura de decisión/condición

```
IF (A>0 && C==1)
        X = X + 1
IF (B==3 || D<0)
        Y=0
END
```

TC1	TC2
A=5 C=1 B=3 D=-3	A=0 C=5 B=5 D=5

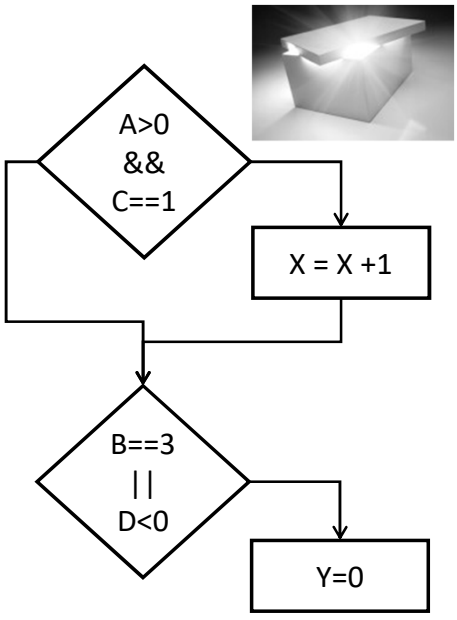


Caja Blanca

Cobertura de múltiple

```
IF (A>0 && C==1)
        X = X + 1
IF (B==3 || D<0)
        Y=0
END
```

TC1	TC2	TC3	TC4
A=5 C=1 B=3 D=-3	A=0 C=1 B=3 D=5	A=5 C=8 B=7 D-3	A=0 C=8 B=7 D=5



Elegir un método

- Cada uno tiene fortalezas y debilidades particulares: un método puede ser bueno para algunas cosas, y no para otras cosas
- El mejor método es no usar un único método. Usar una variedad de técnicas ayudará a un testing efectivo.

39