ILERNA Online

Videotutoría 4: Optimización del SW

Módulo 05: Entornos de Desarrollo

RESUMEN VT03

¿Qué habéis aprendido la VT anterior?

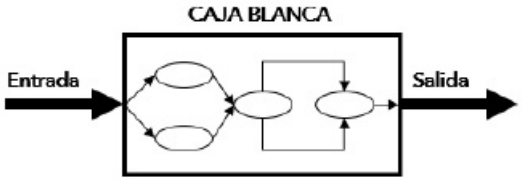
ILERNA Online

Objetivos VT 4

- Teoría de grafos

Pruebas unitarias

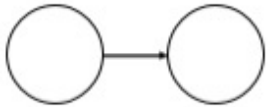
- Pruebas estructurales o de caja de cristal. Su funcionamiento se basa en un exhaustivo examen de los detalles procedimentales del código.



- Se les conoce como prueba de comportamiento. Se realiza sobre la interfaz sin necesidad de conocer la estructura del programa ni cómo funciona.

Prueba del camino básico (caja blanca) (para optimizar nuestro código).

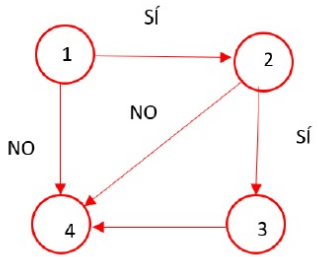
- Es una técnica que permite al desarrollador obtener la medida de la complejidad de nuestro sistema.
- Puede usar esta medida para la definición de un conjunto de caminos de ejecución.
- Para obtener esta medida de complejidad utilizaremos la técnica de representación de grafo de flujo.
- Un nodo es cualquier tipo de sentencia en programación. Por ejemplo, cont=con+1; → nodo 1, cont=0 → nodo 2. Aunque se pueden representar en un mismo nodo ya que son sentencias secuenciales (bucles no) para optimizar estos gráficos.



Secuencial

El método main () {cont=0; cont= cont+1; suma(int a, int b);
Se podría representar en un mismo nodo por ser una llamada secuencial de código.

IF condición AND



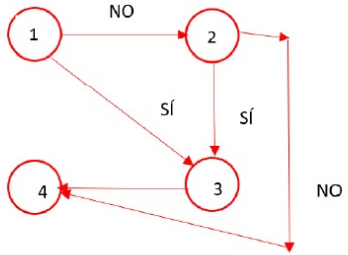
1 2

IF (A && B) {

3 // código

4 }

IF condición OR



1 2

IF (A || B) {

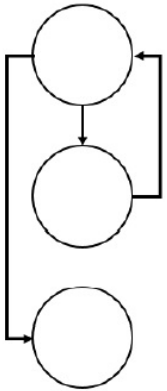
3 // código

4 }

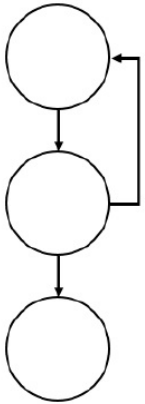
El nodo 3 podría ser un bloque de código secuencial.

Prueba del camino básico (caja blanca)

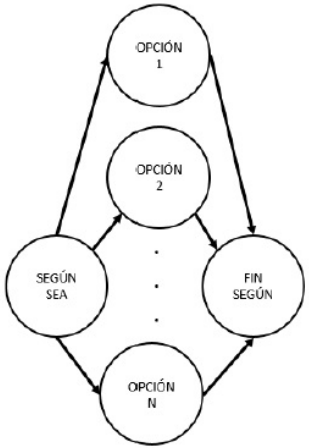
- Es una técnica que permite al desarrollador obtener la medida de la complejidad de nuestro sistema.
- Puede usar esta medida para la definición de un conjunto de caminos de ejecución.
- Para obtener esta medida de complejidad utilizaremos la técnica de representación de grafo de flujo.



WHILE



DO…WHILE



SWITCH

ILERNA Online

Complejidad ciclomática (nos sirve para optimizar)

- Métrica del software que nos proporciona una medida cuantitativa de la complejidad lógica de un programa. Nos establecerá el número de casos de prueba que deberán ejecutarse para que las sentencias sean ejecutadas al menos una vez.
- La complejidad ciclomática **V(G)** se podrá calcular de 3 formas:
  - 1) V(G) = Número de regiones del grafo. Los dos triángulos y el área externa, el cuadrado que forma el conjunto.
  - 2) V(G) = Aristas – Nodos + 2
  - 3) V(G) = Nodos predicado + 1

Nodo predicado es aquel del que salen 2 o más flechas. En la diapositiva 5, en *IF condición AND*, lo serían los números 1 y 2.

Los 3 puntos deben sumar lo mismo, esto es, 3.

Complejidad ciclomática	Evaluación del riesgo
Entre 1 y 10	Programas o métodos sencillos, sin mucho riesgo
Entre 11 y 20	Programas o métodos más complejos, riesgo moderado
Entre 21 y 50	Programas o métodos complejos, alto riesgo
Mayor que 50	Programas o métodos no testeables, muy alto riesgo

Cálculo del camino básico (el mínimo camino para pasar al menos una vez por todos los nodos y todas las aristas)

Esta complejidad ciclomática me va a determinar cuál es la cota inferior del número de pruebas que yo tengo que realizar, como mínimo, todos los caminos existentes al menos una vez por cada uno y una vez por cada arista del grafo.

Resumen en video

Prueba del Camino Básico (Pruebas de Caja Blanca)

ILERNA Online

Ejemplo

```
1:      do while queden registros
          leer registros;
2:        if campo 1 del registro = 0
3:          then procesar registro;
                guardar en buffer;
                incrementar contador;
4:          elseif campo 2 del registro = 0
5:            then reinicializar contador;
6:            else procesar registro;
                guardar en archivo;
7a:         endif
        endif
7b:   enddo
8:0: end
```

PAC de desarrollo

Es un while, no un do while, si no tendríamos el do delante del while.

```
1  WHILE NOT final DO
     2   leer
     3   IF campo1=0 THEN
           4  procesar()
           5  incrementar_conta().
     6   ELSE IF campo1=1 THEN
     7     reinic. conta.()
         ELSE
     8     procesar()
     9   END IF
10  END WHILE
11
```

DESCRIPCIÓN

- Realiza el diagrama de grafo del siguiente pseudocódigo.
- Calcula su complejidad ciclomática y representa el número de caminos mínimo

pseudocodigo.jpg

INSTRUCCIONES DE ENTREGA

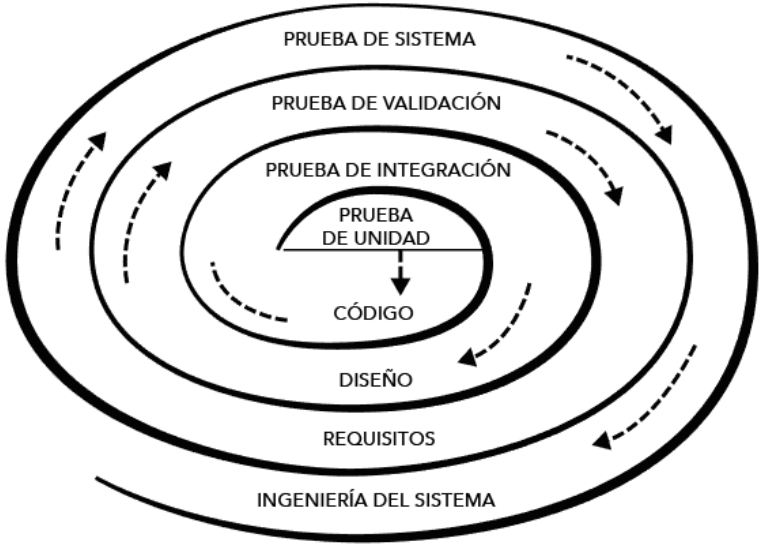
Sigue las siguientes instrucciones para realizar correctamente la entrega del proyecto:

- El fichero en formato **.pdf** debe de llevar la siguiente nomenclatura: **PAC_UF2_ApellidoNombre.pdf**
- Para que el ejercicio sea corregido, este debe entregarse obligatoriamente en la tarea. No se corregirán proyectos entregados por otros medios, como los adjuntos de los comentarios de la tarea o los mensajes privados.
- Se tendrá en cuenta la presentación

FECHA ENTREGA 3 ABRIL

ILERNA Online

Estrategias de prueba SW

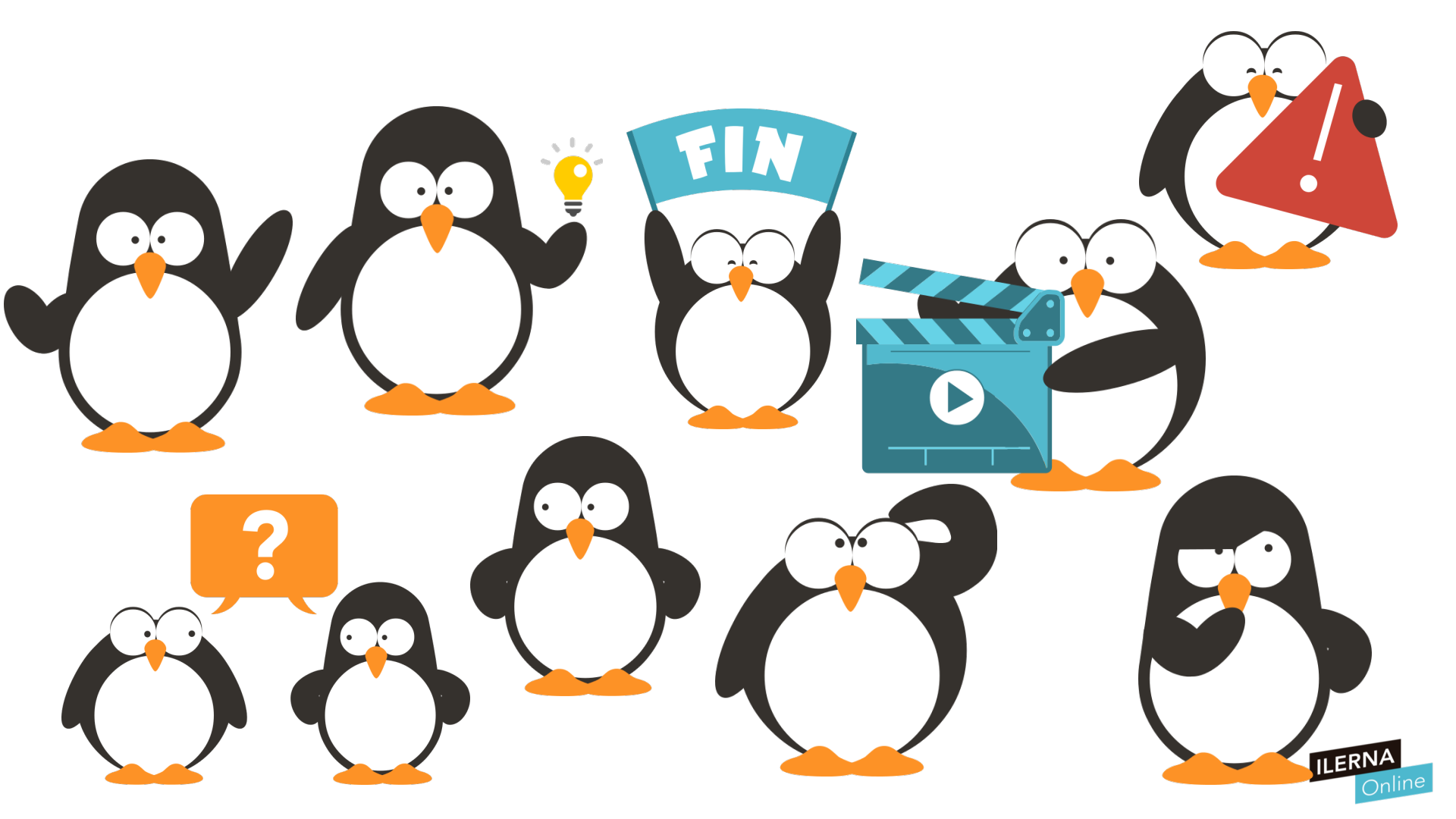


Estrategias de prueba SW

- Pruebas de unidad: En esta prueba vamos a comprobar cada módulo para eliminar cualquier tipo de error en la interfaz o en la lógica interna. Utiliza ambas técnicas, tanto la prueba de la caja negra como la de la blanca. P.ej. JUNIT
- Pruebas de integración:
    - 1) Integración no incremental o big bang. Comprobación de cada módulo por separado y después se prueba de forma conjunta. Se detectan muchos errores y la corrección es difícil.
    - 2) Integración incremental. En este caso el programa se va creando y probando en pequeñas secciones por lo que localizar los fallos es más sencillo. En esta integración podemos optar por dos estrategias:
    - a. Ascendente. Se comienza con los módulos más bajos del programa.
    - b. Descendente. Se empieza por el módulo principal descendiendo por la jerarquía de control.

Estrategias de prueba SW

- Pruebas de Validación: Conseguiremos la prueba de validación cuando el programa funcione de acuerdo con las expectativas expuestas por el cliente
    - Prueba Alfa: realizada por el cliente o usuario en el lugar de desarrollo. Usará el programa bajo la observación del desarrollador que irá registrando los errores.
    - Prueba Beta: realizada por los usuarios finales en su lugar de trabajo sin la presencia del desarrollador. En este caso será el usuario el que registre los errores y se los comunique al desarrollador para que realice las modificaciones correspondientes
- Pruebas de sistema:
    - Prueba de recuperación: se fuerza el fallo del software y que la recuperación se realice correctamente.
    - Prueba de seguridad: se comprueba que el sistema esté protegido frente a acciones ilegales.
    - Prueba de resistencia (Stress). se realizan acciones que requieran una gran cantidad de recursos.


FIN
?
ILERNA
Online