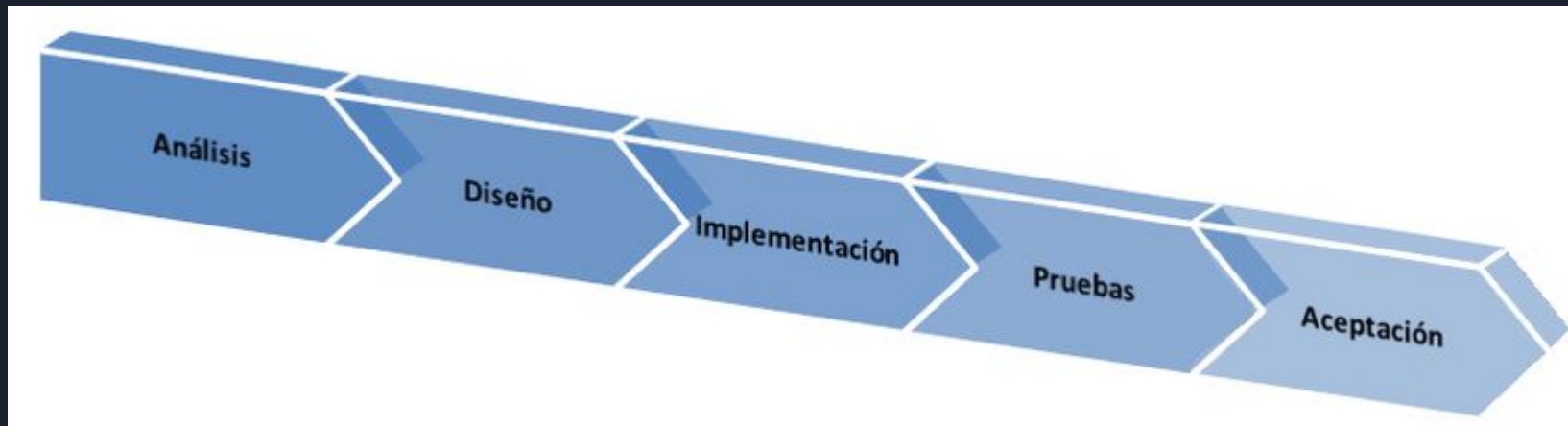


Diseño y realización de pruebas

Introducción

Las **pruebas** son la **cuarta etapa** del ciclo de vida software y tienen el objetivo de **verificar y validar** un producto antes de comenzar su vida útil. De una manera más coloquial, las pruebas consisten en **probar** la aplicación creada.





Técnicas de diseño de casos de prueba

Caso de prueba: conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollado para conseguir un objetivo. Para llevarlo a cabo:

- **Precondiciones**
- **Valores de entrada**
- **Comportamiento esperado**

Tras esto, se observa qué ocurre y si se ha cumplido el resultado previsto o no.



Técnicas de diseño de casos de prueba

Ejemplo: Eliminar a un usuario del sistema

- Precondiciones: El usuario existe
- Valores de entrada: El usuario a borrar

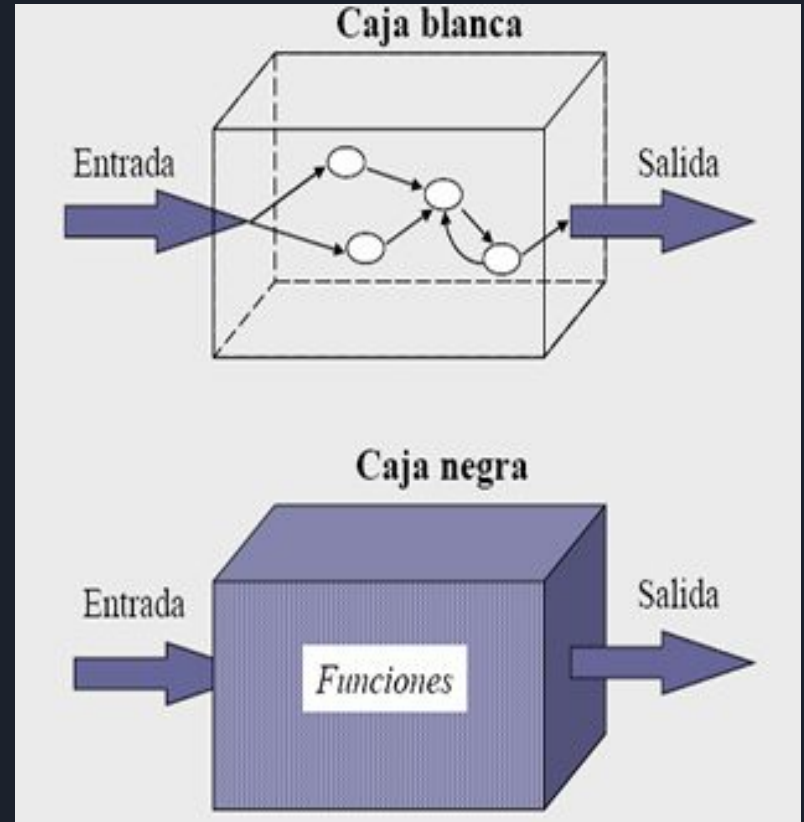
Resultado esperado: El usuario ya no existe en la base de datos y podemos crear otro usuario con los mismos datos que tenía el usuario eliminado.

¿Cómo sería el caso de prueba para añadir un producto al carrito?

Técnicas de diseño de casos de prueba

Técnicas/Enfoques:

- Caja blanca
- Caja negra





Técnicas de diseño de casos de prueba

Caja blanca: Son pruebas estructurales o de caja de cristal. Se centran en examinar los detalles del procedimiento que sigue el código.

Podemos obtener casos de prueba que:

- Se ejecuten al menos una vez todos los caminos independientes de cada módulo
- Se ejecuten al menos una vez todas las sentencias
- Se ejecuten todas las decisiones lógicas en las partes V o F
- Se ejecuten los bucles dentro de sus límites
- Se utilicen todas las estructuras de datos internas para asegurar su validez



Técnicas de caja blanca

Prueba del camino básico

Es una técnica que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica y usarla como guía para la definición de los caminos de ejecución.

¿Cómo obtenemos la medida de complejidad? **Grafo de flujo**

Para entender el grafo de flujo tenemos que comprender lo que significan los elementos que lo componen.



Técnicas de caja blanca

Prueba del camino básico

Antes de ver los elementos gráficos debemos conocer algunos conceptos:

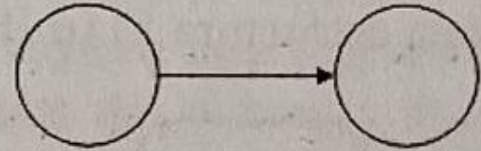
- Los círculos son **nodos** y estos representan una o más sentencias procedimentales
- Las flechas son **aristas** o enlaces y representan el flujo de control
- Una arista siempre termina en un nodo
- Las áreas delimitadas por aristas y nodos son **regiones**. El área exterior del nodo es un área más.
- El nodo que contiene una condición es un **nodo predicado**. De él salen dos o más aristas.

A continuación se muestran las estructuras de control que se usan para navegar por el grafo y el código.

Técnicas de caja blanca

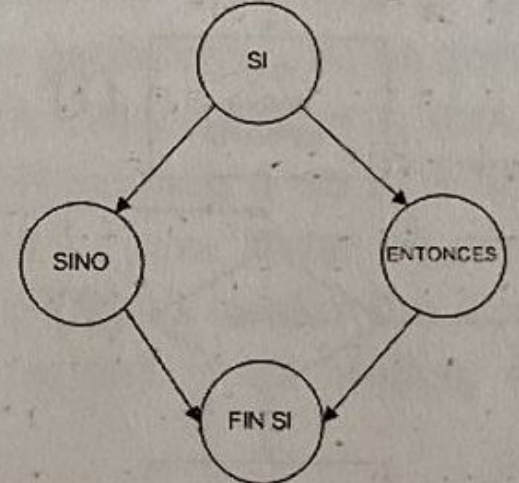
SECUENCIAL

Instrucción 1
Instrucción 2
.....
Instrucción n



CONDICIONAL

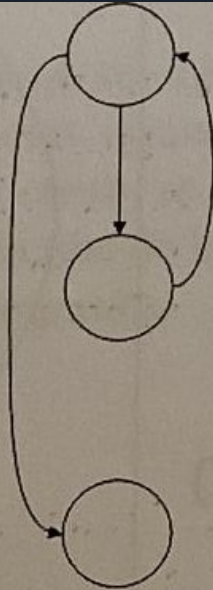
Si <condición> **Entonces**
 <Instrucciones>
Si no
 <Instrucciones>
Fin si



Técnicas de caja blanca

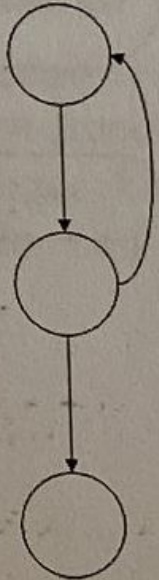
HACER MIENTRAS

Mientras <condición> **Hacer**
 <instrucciones>
Fin mientras



REPETIR HASTA

Repetir
 <instrucciones>
Hasta que <condición>



Técnicas de caja blanca

CONDICIONAL MÚLTIPLE

Según sea <variable> **Hacer**

Caso opción 1:

<Instrucciones>

Caso opción 2:

<Instrucciones>

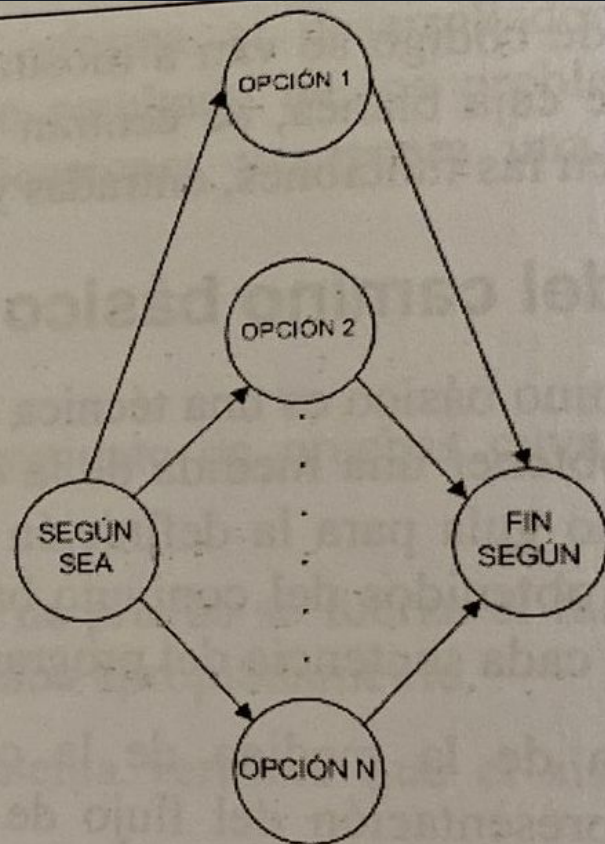
Caso opción 3:

<Instrucciones>

Otro caso:

<Instrucciones>

Fin según



Técnicas de caja blanca

Prueba del camino básico

Ejemplo:

```
int c = 0;
int par = 0;
int impar = 0;
Scanner tecla = new Scanner( source: System.in );

while(c!=10) {

    System.out.print( s: "Dame un número entero: " );

    if(tecla.nextInt()%2==0) {
        System.out.print( s: " PAR " );
        par++;
    }
    else{
        System.out.print( s: " IMPAR " );
        impar++;
    }
    c++;
}

System.out.print(" Hay" +par+" numeros pares y " +impar+ "impares" );
```



Técnicas de caja blanca

Prueba del camino básico

Los caminos que podemos elegir son:

- Camino 1: 1-2-7
- Camino 2: 1-2-3-4-6-2-7
- Camino 3: 1-2-3-5-6-2-7



Técnicas de caja blanca

Prueba del camino básico

Obtención de los casos de prueba

Caminos	Caso de prueba	Resultado esperado
1	Algún valor de C que no cumpla la condición de $C \neq 10$. $C = 10$	Muestra por pantalla el número de pares e impares
2	Algún valor de C que cumple la condición $C \neq 10$. Escoger un valor de tecla que no cumpla $tecla \% 2 = 0$ $C = 1$, tecla = 3	Cuenta un impar más
3	Algún valor de C que cumple la condición $C \neq 10$. Escoger un valor de tecla que cumpla $tecla \% 2 = 0$ $C = 2$, tecla = 4	Cuenta un par más



Técnicas de caja blanca

Ej1: Diseña el grafo de flujo, los caminos y los casos de prueba del siguiente código.

```
static void visualizarMedia (float x, float y){  
    float resultado = 0;  
    if(x<0 || y<0)  
        System.out.println("X e Y deben ser positivos");  
    else{  
        resultado = (x + y)/2;  
        System.out.println("La media es: "+resultado);  
    }  
}
```

```
public void name(String curso, string sexo, int nota) {  
    string c;  
  
    while(ReadAlumnos()){  
        c=visualizarCurso();  
  
        while(curso == ){  
            if(sexo=="H")  
                numHombres++;  
            else  
                numMujeres++;  
        }  
    }  
}
```

Realiza el grafo de flujo, los caminos básicos y la tabla correspondiente del siguiente código.



Técnicas de caja blanca

Complejidad ciclomática $V(G)$

Es la métrica del software que nos proporciona una medida cuantitativa de la complejidad lógica de un programa.

En el método de prueba del “Camino básico” establece el número de **caminos independientes**. Por tanto, establece el número de casos de prueba que se deben ejecutar para asegurar que cada sentencia se ejecuta al menos una vez.

Se puede calcular de la siguiente manera.

$V(G)$: n° de regiones del grafo.

$V(G)$: Aristas - Nodos + 2

$V(G)$: Nodos predicado + 1



Técnicas de caja blanca

Complejidad ciclomática $V(G)$

$V(G)$	Evaluación del riesgo
[1-10]	Programas o métodos sencillos, sin mucho riesgo
[11-20]	Programas o métodos más complejos, riesgo moderado
[21-50]	Programas o métodos complejos, alto riesgo
>50	Programas o métodos no testeables, riesgo muy alto



Técnicas de caja blanca

Calcula la complejidad ciclomática de los ejemplos realizados hasta ahora.



Técnicas de diseño de casos de prueba

Caja negra: son las pruebas de comportamiento que se llevan a cabo sobre la interfaz del SW.

No necesitamos conocer la estructura interna del programa ya que se pretende obtener casos de prueba que demuestren que las **funciones del programa** producen las salidas esperadas a partir de las entradas proporcionadas.

Intentamos encontrar distintos tipos de errores:

- Funcionalidades incorrectas o ausentes: No nos deja acceder a nuestra cuenta con los datos correctas
- De interfaz: no se nos muestra la interfaz correcta en alguna situación
- En estructuras de datos: No estamos guardando los datos necesarios
- De rendimiento: la aplicación consume demasiados recursos
- De inicialización y finalización



Métodos de prueba de caja negra

Partición/Clases de equivalencia: divide los valores de los campos de entrada de un programa en clases de equivalencia.

Identificar las clases de equivalencia: examinamos las condiciones de entrada y dividimos en grupos.

Tipos de clases de equivalencia:

- Válidas: valores de entrada válidos.
- No válidas: valores de entrada no válidos.

Las clases de equivalencia se definen según una serie de directrices.

Métodos de prueba de caja negra

Condiciones de entrada	Nº de Clases de equivalencia válidas	Nº de Clases de equivalencia no válidas
1. Rango	1 CLASE VÁLIDA Contempla los valores del rango	2 CLASES NO VÁLIDAS Un valor por encima del rango Un valor por debajo del rango
2. Valor específico	1 CLASE VÁLIDA Contempla dicho valor	2 CLASES NO VÁLIDAS Un valor por encima Un valor por debajo
3. Miembro de un conjunto	1 CLASE VÁLIDA Una clase por cada uno de los miembros del conjunto	1 CLASE NO VÁLIDA Un valor que no pertenece al conjunto
4. Lógica	1 CLASE VÁLIDA Una clase que cumpla la condición	1 CLASE NO VÁLIDA Una clase que no cumpla la condición



Ejemplo clases de equivalencia.

Se va a realizar una entrada de datos de un empleado por pantalla gráfica, se definen 2 campos de entrada y una lista para elegir el oficio. La aplicación acepta los datos de esta manera.


- Empleado: nº de 3 dígitos que no empiece por 0.
- Departamento: en blanco o un nº de 2 dígitos.
- Oficio: Analista, Diseñador, Programador o Elegir Oficio.

Si la entrada es correcta, el programa asigna un salario (que se muestra por pantalla) a cada empleado según estas normas:

- S1: El Oficio es Analista se asigna 2.500
- S2: El Oficio es Diseñador se asigna 1.500
- S3: El Oficio es Programador se asigna 2.000

Si la entrada no es correcta el programa muestra un mensaje indicando la entrada que es incorrecta.

- ER1 si el Empleado es incorrecto
- ER2 si el Departamento no es correcto
- ER3 si no se ha elegido Oficio.



Condición de entrada	Clases de equivalencia	Clases Válidas	COD	Clases no Válidas	COD
Empleado	Rango	100 >= Empleado <= 999	V1	Empleado < 100 Empleado > 999	NV1 NV2
Departamento	Lógica (puede estar o no)	En blanco	V2	No es un número	NV3
	Valor	Cualquier número de 2 dígitos	V3	Número de más de 2 dígitos Numero de menos de 2 dígitos	NV4 NV5
Oficio	Miembro de un conjunto	Oficio="Programador"	V4	Oficio="Elige oficio"	NV8
		Oficio="Analista"	V5		
		Oficio="Diseñador"	V6		

CASO DE PRUEBA	Clases de equivalencia	CONDICIONES DE ENTRADA			Resultado esperado
		Empleado	Departamento	Oficio	
CP1	V1, V3, V4	200	20	Programador	S3
CP2	V1, V2, V5	250		Analista	S1
CP3	V1, V3, V6	450	30	Diseñador	S2
CP4	V1, V2, V4	220		Programador	S3
CP5	NV1, V3, V6	90	35	Diseñador	ER1
CP6	V1, NV3, V5	100	AD	Analista	ER2
CP7	V1, V2, NV8	300		Elige oficio	ER3
CP8	V1, NV4, V6	345	123	Diseñador	ER2
...	...				



Actividad

Tenemos la siguiente función en java

```
public String parImpar(int num)
{
    if(num %2==0)
        cad="Par";
    else
        cad="Impar";
    return cad;
}
```

- Realiza una tabla donde especifiques la condición de entrada, las clases de equivalencia, las clases válidas, un código para identificarlas, las clases no válidas y un código para especificarlas.
- Realiza otra tabla donde especifiques el número de caso de prueba, las clases de equivalencia que intervienen, la condición de entrada y el resultado esperado.



Actividad: clases de equivalencia

Vamos a inventarnos un lenguaje que tenga las siguientes características:

No debe tener más de 15 ni menos de 5 caracteres.

El juego de caracteres utilizables es:

- letras (Mayúsculas y minúsculas)
- dígitos (0,9)
- Guión (-). El guión no puede estar ni al principio ni al final y no pueden haber consecutivos.

Debe contener al menos un carácter alfabético.

No puede ser una de las palabras reservadas del lenguaje



Métodos de prueba de caja negra

Análisis de valores límite: Se basa en que los errores tienden a producirse con más probabilidad en los extremos de los campos de entrada.

Esta técnica complementa a la anterior de manera que:

- Rango de valores: casos de prueba para los límites del rango y los valores justo por encima y por debajo de este.
- Número de valores (Por ejemplo escribir de 2 a 10 caracteres): probaremos para 2, para 10, para 1 y para 11 datos de entrada.
- Estructuras de datos internas (Por ejemplo un array de 5 valores): probaremos que lea 1, 5, 6, y 0.



Prueba de unidad

Se prueba cada módulo con el objetivo de eliminar errores en la interfaz y en la lógica interna.

Se utilizan técnicas de caja blanca y caja negra.

Se realizan pruebas sobre:

- Correcto flujo de información en el interfaz del módulo.
- Estructuras de datos locales.
- Condiciones límite.
- Todos los caminos independientes de la estructura de control para asegurar que se ejecutan todas las sentencias al menos una vez.

Algunas herramientas para pruebas unitarias: JUnit, CPPUnit, PHPUnit.



Prueba de integración

Se observan las interacciones de los módulos.

Dos enfoques:

- Integración no incremental o big bang: se prueba cada módulo por separado y se combinan todos de una vez.
- Integración incremental: el programa completo se va construyendo y probando en pequeños fragmentos.



Prueba de validación

Se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente definidas en el documento de especificación de requisitos (ERS).

Se llevan a cabo pruebas de caja negra.

Técnicas a realizar:

- Prueba **Alfa**: llevado a cabo por el cliente o usuarios en el lugar de desarrollo.
- Prueba **Beta**: Llevada a cabo por el usuario final en su lugar de trabajo.



Prueba del sistema

Formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software.

- Prueba de recuperación: se fuerza al fallo del software y se verifica que la recuperación se lleva a cabo apropiadamente.
- Prueba de seguridad: verificar que el sistema está protegido contra accesos ilegales.
- Prueba de resistencia (Stress): sistema o situaciones que requieren una gran cantidad de recursos.



Documentación de pruebas

Conjunto de documentos que se pueden producir durante el proceso de prueba según el estándar IEEE 829-1998.

- Plan de pruebas:
- Especificaciones de prueba: formada por tres tipos de documentos
 - Especificación del diseño de prueba.
 - Especificación de los casos de prueba.
 - Especificación de los procedimientos de prueba.
- Informes de prueba:
 - Informe que identifica los elementos que están siendo probados.
 - Registro de las pruebas.
 - Incidentes de pruebas.
 - Resumen de las actividades de prueba.