# UT05. DISEÑO Y REALIZACIÓN DE PRUEBAS

Entornos de Desarrollo

1DAW – C.I.F.P. Carlos III - Cartagena

# Índice

# 1. Planificación de las pruebas

# Planificación de las pruebas

- Durante todo el proceso de desarrollo de software, es necesario realizar un conjunto de pruebas →**software correcto**

  →software que cumple con las especificaciones impuesta por el usuario.
- **Errores** del proceso de desarrollo de software
  - Una incorrecta especificación de los objetivos
  - Errores producidos durante el proceso de diseño
  - Errores que aparecen en la fase de desarrollo.

# Planificación de las pruebas

- Tareas en la **fase de pruebas:**
  - **Verificación**: comprobación que un sistema o parte de un sistema, cumple con las condiciones impuestas. ¿Se está construyendo correctamente?
  - **Validación**: proceso de evaluación del sistema o de uno de sus componentes, para determinar si satisface los requisitos especificados.
- Es necesario implementar una estrategia de pruebas.
  - **Prueba de unidad**, donde se analizaría el código implementado
  - **Prueba de integración**, donde se prestan atención al diseño y la construcción de la arquitectura del software.
  - **Prueba de validación**, comprueba que el sistema construido cumple con lo establecido en el análisis de requisitos de software.
  - **Prueba de sistema**, verifica el funcionamiento total del software y otros elementos del sistema.

# 2. Tipos de prueba

# Tipos de prueba

- **Caso de prueba-** conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollado para conseguir un objetivo particular o condición de prueba.
  - Definir las precondiciones y postcondiciones.
  - Identificar los valores de entrada
  - Conocer el comportamiento  que debería tener el sistema ante dichos valores

# Tipos de prueba

- **Prueba de la Caja Negra** (Black Box Testing):
  - Prueba mediante su interfaz externa, sin la implementación de la aplicación.
  - Comprueba que los resultados de la ejecución de la aplicación, son los esperados, en función de las entradas que recibe.
    - No es necesario conocer ni la estructura, ni el funcionamiento interno del sistema.
    - Solo se conocen las entradas a recibir por la aplicación, y las salidas que les correspondan, pero no se conoce el proceso mediante el cual la aplicación obtiene esos resultados.

# Tipos de prueba

# Tipos de prueba

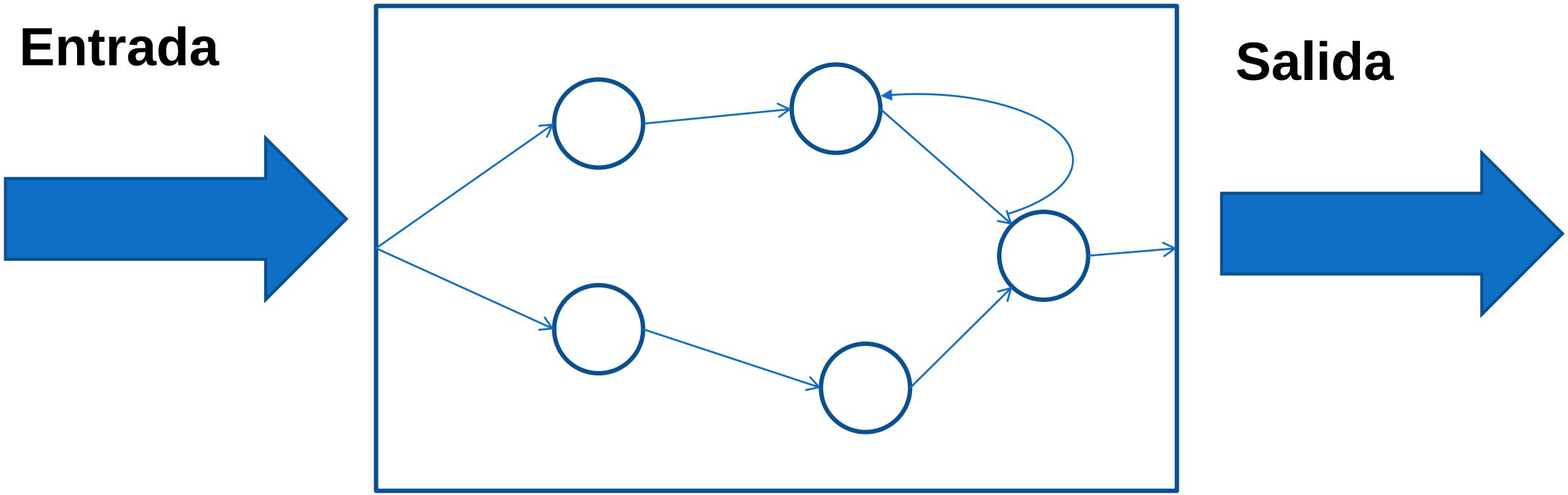
- Pruebas de comportamiento
- **Categorías de los errores** que se encuentran
  - Funcionalidades incorrectas o ausentes
  - Errores de interfaz
  - Errores en estructuras de datos o en accesos a BD externas
  - Errores de rendimiento
  - Errores de inicialización y finalización

# Tipos de prueba

- **Prueba de la Caja Blanca** (White Box Testing):
  - Prueba de la aplicación desde dentro, usando su lógica de aplicación.
  - Análisis y prueba directamente el código de la aplicación, con conocimiento específico del código.

# Tipos de prueba

## Prueba de caja blanca



# Tipos de prueba

- **Caja blanca**
  - Se pueden obtener casos de prueba que:
    - Garanticen que se ejecutan al menos una vez **todos los caminos** independientes de cada módulo
    - Ejecuten **todas las sentencias** al menos una vez
    - Ejecuten **todas las decisiones lógicas** en su parte verdadera y en su parte falsa
    - Ejecuten **todos los bucles** en sus límites
    - Utilicen **todas las estructuras de datos internas** para asegurar su validez

- Tipos de prueba
  - Funcionales
  - Estructurales
  - Regresión

# Funcionales

- Son pruebas de la **caja negra**.
- **Objetivo**: verificar una acción específica o funcional dentro del código de una aplicación.
- ¿Puede el usuario hacer esto?
- ¿Funciona esta utilidad de la aplicación?
- Analiza las entradas y las salidas de cada componente, **verificando que el resultado es el esperado**, sin preocuparnos por la estructura interna del mismo.

# Funcionales

- Tipos de pruebas funcionales
  - Particiones o clases de equivalencia
  - Análisis de valores límite
  - Pruebas aleatorias

# Funcionales. Particiones o clases de equivalencia

- **Particiones o clases de equivalencia**
  - Definir un buen caso de prueba.
    - Cada clase debe cubrir el máximo número de entradas
    - Debe tratarse el **dominio de valores** de entrada dividido en un número finito de clases de equivalencia que cumplan la siguiente propiedad:
      - → "*La prueba de un valor representativo de una clase permite suponer razonablemente que el resultado obtenido será el mismo que el obtenido probando cualquier otro valor de la clase*"
  - Pasos.
    - **Identificación** de los casos de prueba
    - **Creación** de los casos de prueba

# Funcionales. Particiones o clases de equivalencia

- **Particiones o clases de equivalencia**
  - Identificar las clases de equivalencia
    - Identificación de las **condiciones de las entradas** del programa
    - Identificación de las **clases**
      - → De datos válidos
      - → De datos no válidos
    - **Reglas** que ayudan a identificar clases.
      - → **Rango**➔clase válida (rango) y clase no válidas (extremos del rango)
      - → **Número de valores**➔ clase válida (rango) y clase no válidas (extremos del rango)
      - → **Boolean** o "debe ser"➔ clase válida (se cumple) y clase no válida (no se cumple)
      - → **Conjunto de valores**➔ clase válida (uno de los valores) y clase no válida (cualquier otro caso)
      - → Dividir en **subclases** si se llega a la conclusión de que tiene diferentes tratamientos

# Funcionales. Particiones o clases de equivalencia

- **Ejemplo**
  - Aplicación bancaria donde el operador debe aportar un **código**, un nombre de **operación** y una **orden** que disparará una serie de funciones bancarias
  - Especificación
    - **Código área**: número de 3 dígitos que no empieza por 0 ni por 1.
    - **Nombre de identificación**: 6 caracteres.
    - **Órdenes** posibles: cheque, depósito, pago factura, retirada de fondos.

# Ejemplo clases de equivalencia

- **Código**
  - Número
    - **Clase válida:** número
    - **Clase no válida:** no número
  - División de número
    - Subclase **válida** (200-999)
    - Dos subclases **no válidas**
      - → Menor de 200
      - → Mayor de 999
- **Nombre id**
  - Clase **válida:** 6 car.
  - Clases **no válidas**
    - Más de 6 car.
    - Menos de 6 car.
- **Orden**
  - Clase **válida** para cada orden: 4 en total.
  - Clase **no válida:** p.e. divisas

# Ejemplo clases de equivalencia

| Condición de entrada | Clases válidas | Clases no válidas |
|---|---|---|
| Código área | (1) 200<=código<=999 | (2) Código < 200<br>(3) Código > 999<br>(4) No es número |
| Nombre de operación | (5) Seis caracteres | (6) Menos de 6 caracteres<br>(7) Más de 6 caracteres |
| Orden | (8) Cheque<br>(9) Depósito<br>(10) Pago factura<br>(11) Retirada de fondos | (12) Ninguna orden válida |

# Ejemplo clases de equivalencia

- **Casos válidos**
  - 300 nómina depósito **(1) (5) (9)**
  - 400 viajes cheque **(1) (5) (8)**
  - 500 coche pago-factura **(1) (5) (10)**
  - 600 comida retirada de fondos **(1) (5) (11)**
- **Casos no válidos**
  - 180 Viajes Pago-factura **(2) (5) (10)**
  - 1032 Nómina Depósito **(3) (5) (9)**
  - XY Compra Retirada-fondos **(4) (5) (11)**
  - 350 A Depósito **(1) (6) (9)**
  - 450 Regalos Cheque **(1) (7) (8)**
  - 550 Casa &%4 **(1) (5) (12)**

# Ejemplos

1. Se va a realizar una entrada de datos de un empleado por pantalla gráfica, se definen 3 campos de entrada y una lista para elegir el oficio. La aplicación acepta los datos de esta manera:
   - Empleado: número de tres dígitos que no empiece por 0
   - Departamento: en blanco o número de dos dígitos
   - Oficio: analista, diseñador, programador o elige oficio
- Si la entrada es correcta, el programa asigna un salario (que se muestra en pantalla) a cada empleado según estas normas:
  - S1 si el oficio es analista se asigna 2500
  - S2 si el oficio es diseñador se asigna 1500
  - S3 si el oficio es programador se asigna 2000
- Si la entrada no es correcta, el programa muestra un mensaje indicando la entrada incorrecta
  - ER1 si el empleado no es correcto
  - ER2 si el departamento no es correcto
  - ER3 si no se ha elegido oficio

# Ejemplos

**2.** Tenemos una función Java que recibe un número entero y devuelve una cadena con el texto “Par” si el número recibido es par, o “Impar” si el número es impar.

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