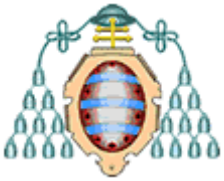
Pruebas y Calidad del Software

Ingeniería del Software

José Ramón de Diego



Ejercicio previo

- **Enunciado**: Tenemos un programa que
  - Lee tres enteros de un fichero
  - Los tres enteros representan los lados de un triángulo
  - Imprime un mensaje indicando el tipo de triángulo
    - Escaleno
    - Isósceles
    - Equilátero

- **Problema**: Escribir el conjunto de casos de prueba adecuados para probar el programa anterior

Pruebas y Calidad del software

- **Pruebas = Ejecutar** el software con la intención de descubrir fallos.
- Tienen el objetivo de mejorar la **calidad** del software.
  - Dan confianza en el producto.
- ¿Qué se entiende por software de **calidad**?
  - El que cumple los requisitos (documentados o implícitos).
  - Dentro de los límites de tiempo y coste.

Características de calidad

Fuente: Norma ISO 9126.

- Funcionalidad
- Fiabilidad
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad

En la asignatura nos vamos a centrar en pruebas funcionales.

Definiciones de prueba de software

- IEEE Standard 610.12-1990: “The process of:
  - **operating a system** or component
  - under **specified conditions**,
  - observing or recording **the results**,
  - and making an **evaluation** of some aspect of the system or component”.
- Otras definiciones (consultar):
  - Kaner: http://www.kaner.com/pdfs/QAIExploring.pdf
  - ISTQB: http://www.istqb.org/downloads/glossary-current.pdf
- La prueba NO es: verificar que el programa se ejecuta correctamente.

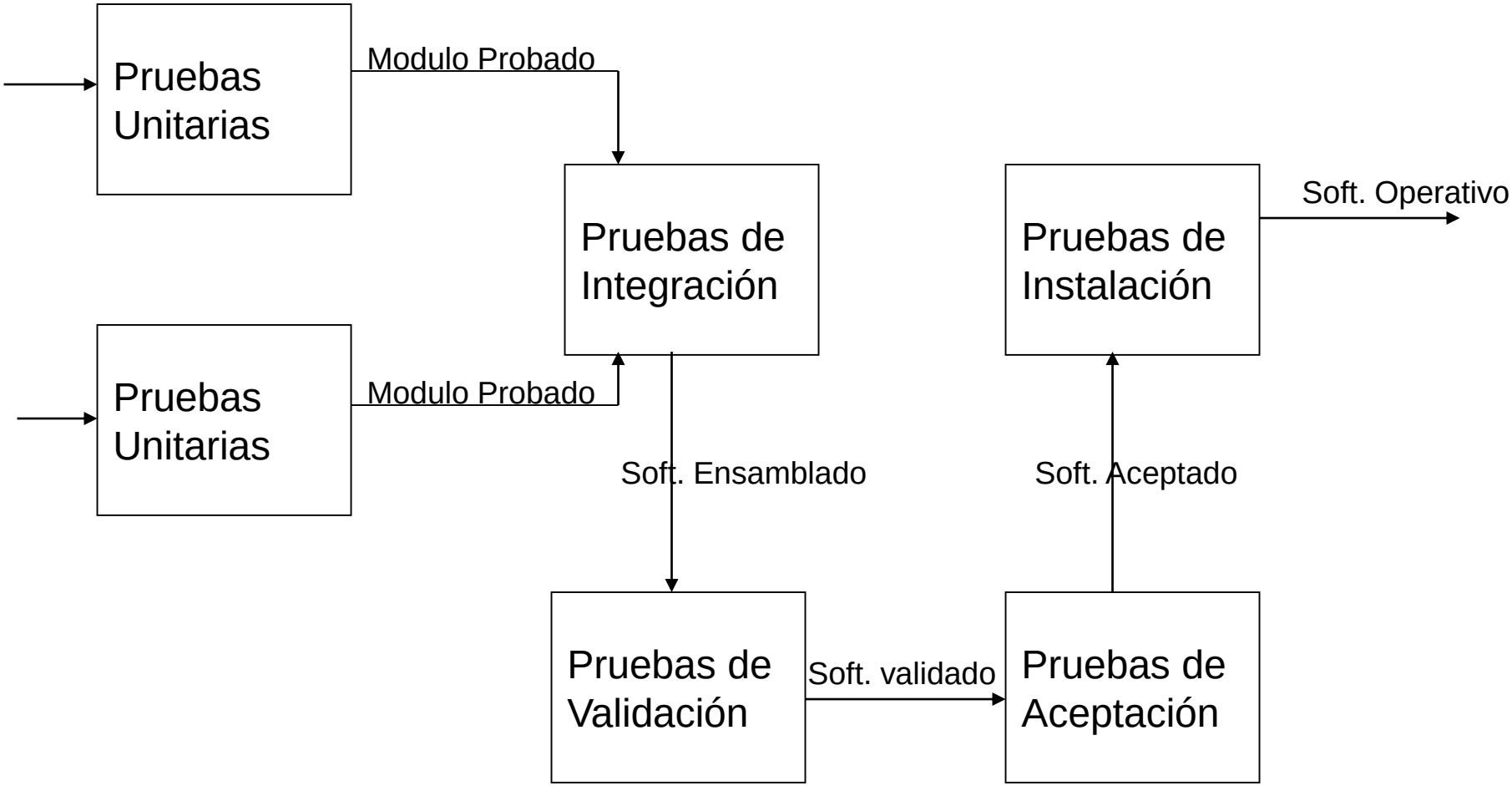
Objetivos de la prueba de software

- Descubrir fallos (ya está dicho)
  - … por tanto una prueba **tiene ÉXITO si descubre un fallo** no detectado,
  - … y una buena prueba es aquella que tiene una **alta probabilidad** de descubrir un fallo no encontrado hasta entonces.

Fallos y defectos

- Un fallo es la **diferencia** entre el comportamiento **esperado** en el software y el **observado**.
- Para corregir un fallo debe encontrarse el **defecto** (o defectos) que lo causa (depuración).
- Detectar el defecto que causa un fallo no es una tarea trivial.

Proceso


Pruebas Unitarias
Modulo Probado
Pruebas de Integración
Pruebas de Instalación
Soft. Operativo
Pruebas Unitarias
Modulo Probado
Soft. Ensamblado
Soft. Aceptado
Pruebas de Validación
Soft. validado
Pruebas de Aceptación

Caso de prueba

- IEEE Standard 610.12-1990: Un **caso de prueba** (CP) es un conjunto de **entradas, condiciones de ejecución y resultados esperados** desarrollados para un objetivo particular.

Caso práctico: carrito de la compra en tienda virtual de libros.

Objetivo: Probar la acción de introducir un nuevo ejemplar de un libro en el carrito.
Condiciones de la prueba:
El catálogo contendrá, al menos, el siguiente libro: 6-332-72934-0 con precio de 70€. El stock del libro será de 3 unidades.
El carrito estará vacío en el momento de introducir el ejemplar.
Entradas: ISBN = 6-332-72934-0, Unidades = 2.
Salida deseada: El libro se introduce correctamente en el carrito, el precio actual del carrito pasa a ser de 140 € y el stock del libro se actualiza a 1 unidad.

Nota: en esta tienda, el control de stocks es un requisito.

Algunos posibles fallos y defectos

- **Fallo**: El stock del libro no se actualiza.
  Defecto: el stock no se actualiza correctamente si se inserta más de un libro a la vez en el carrito.

- **Fallo**: Se introducen los libros en el carrito con un precio total de 120€.
  Defecto: el subsistema de cálculo de ofertas aplica incorrectamente una oferta preparada para activarse en la próxima quincena.

- **Fallo**: “No se puede incluir el libro en el carrito, no hay stock suficiente.”
  Defecto: el sistema no permite introducir libros en el carrito cuyo stock esté por debajo de un parámetro denominado “StockMinimo”, cuyo valor por defecto es 5.

Caso práctico

- Hemos diseñado una prueba para software del que no disponíamos del código fuente (situación típica).
- ¿Cómo diseñar casos de prueba a partir de código fuente? (pruebas de componentes o unitarias).
- Algunas técnicas que se pueden aplicar [Myers, 2004]:
  - Cobertura de sentencias.
  - Cobertura de decisiones.
  - Cobertura de condiciones.
  - Cobertura de múltiple condición.

```
accion IntroducirProducto (Producto pro, Entero unidades, Carrito car)
    si (pro.codigo € catalogo)
        Error ("El producto no está en el catálogo");
    sino
        si (unidades > pro.stock)
            Error ("No existe disponibilidad");
        sino
            si (pro.codigo € ofertas Y hoy <= ofertas.fecha )
                precio = ofertas[pro.codigo].precio;
            sino
                precio = pro.codigo;
            finsi
            /* Crear item e introducirlo en el carrito */
            Item it.Nuevo(pro.codigo, unidades, precio);
            car.Añadir(it);

            /* Actualizar precio del carrito y stock del producto */
            car.Total = car.Total + unidades * precio;
            pro.stock = pro.stock - unidades;
        finsi
     finsi
finaccion
```

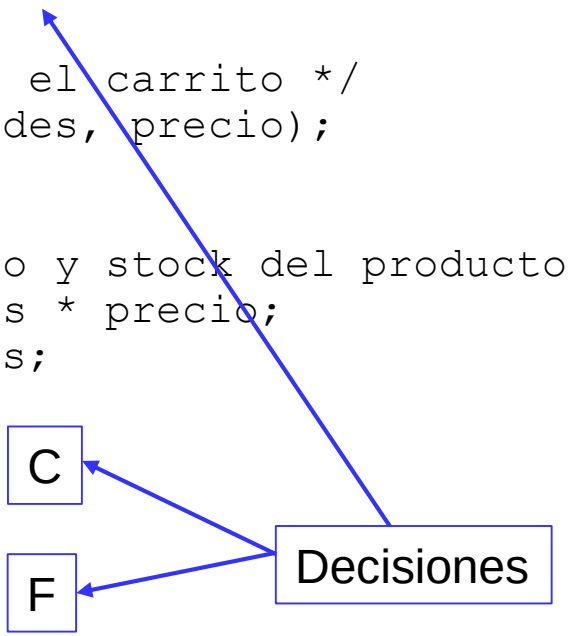
```
accion IntroducirProducto (Producto pro, Entero unidades, Carrito car)
    si (pro.codigo € catalogo)
        Error ("El producto no está en el catálogo");
    sino
        si (unidades > pro.stock)
            Error ("No existe disponibilidad");
        sino
            si (pro.codigo € ofertas Y hoy <= ofertas.fecha )
                precio = ofertas[pro.codigo].precio;
            sino
                precio = pro.codigo;
            finsi
            /* Crear item e introducirlo en el carrito */
            Item it.Nuevo(pro.codigo, unidades, precio);
            car.Añadir(it);

            /* Actualizar precio del carrito y stock del producto */
            car.Total = car.Total + unidades * precio;
            pro.stock = pro.stock - unidades;
        finsi
     finsi
finaccion
```

Sentencias

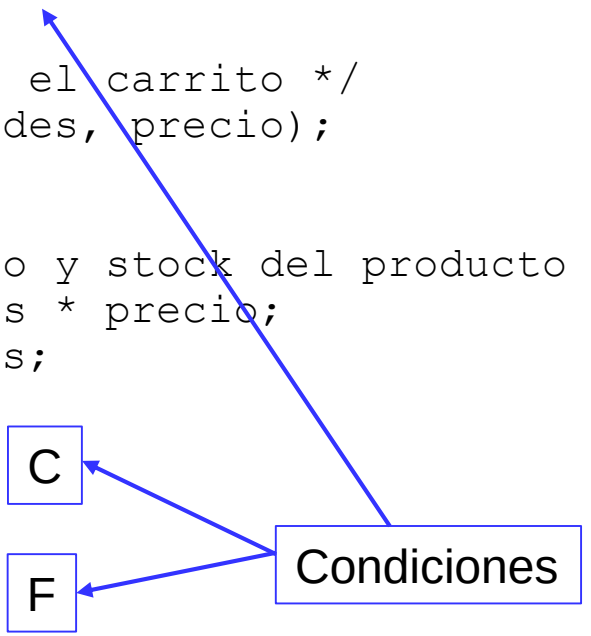
```
accion IntroducirProducto (Producto pro, Entero unidades, Carrito car)
    si (pro.codigo € catalogo)
        Error ("El producto no está en el catálogo");
    sino
        si (unidades > pro.stock)
            Error ("No existe disponibilidad");
        sino
            si (pro.codigo € ofertas Y hoy <= ofertas.fecha )
                precio = ofertas[pro.codigo].precio;
            sino
                precio = pro.codigo;
            finsi
            /* Crear item e introducirlo en el carrito */
            Item it.Nuevo(pro.codigo, unidades, precio);
            car.Añadir(it);

            /* Actualizar precio del carrito y stock del producto */
            car.Total = car.Total + unidades * precio;
            pro.stock = pro.stock - unidades;
        finsi
    finsi
finaccion
```



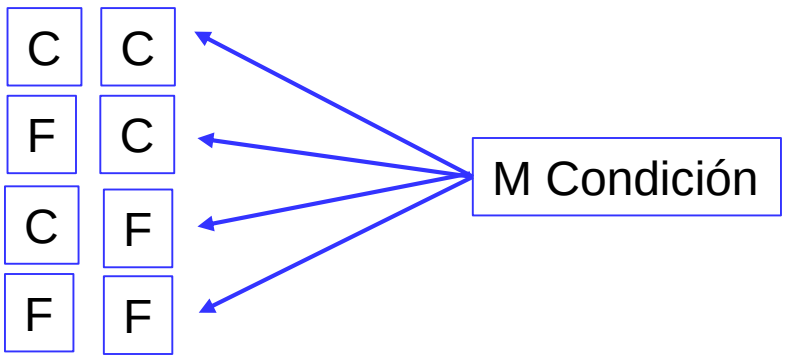
```
accion IntroducirProducto (Producto pro, Entero unidades, Carrito car)
    si (pro.codigo € catalogo)
        Error ("El producto no está en el catálogo");
    sino
        si (unidades > pro.stock)
            Error ("No existe disponibilidad");
        sino
            si (pro.codigo € ofertas Y hoy <= ofertas.fecha )
                precio = ofertas[pro.codigo].precio;
            sino
                precio = pro.codigo;
            finsi
            /* Crear item e introducirlo en el carrito */
            Item it.Nuevo(pro.codigo, unidades, precio);
            car.Añadir(it);

            /* Actualizar precio del carrito y stock del producto */
            car.Total = car.Total + unidades * precio;
            pro.stock = pro.stock - unidades;
        finsi
    finsi
finaccion
```



```
accion IntroducirProducto (Producto pro, Entero unidades, Carrito car)
    si (pro.codigo € catalogo)
        Error ("El producto no está en el catálogo");
    sino
        si (unidades > pro.stock)
            Error ("No existe disponibilidad");
        sino
            si (pro.codigo € ofertas Y hoy <= ofertas.fecha )
                precio = ofertas[pro.codigo].precio;
            sino
                precio = pro.codigo;
            finsi
            /* Crear item e introducirlo en el carrito */
            Item it.Nuevo(pro.codigo, unidades, precio);
            car.Añadir(it);

            /* Actualizar precio del carrito y stock del producto */
            car.Total = car.Total + unidades * precio;
            pro.stock = pro.stock - unidades;
        finsi
    finsi
finaccion
```

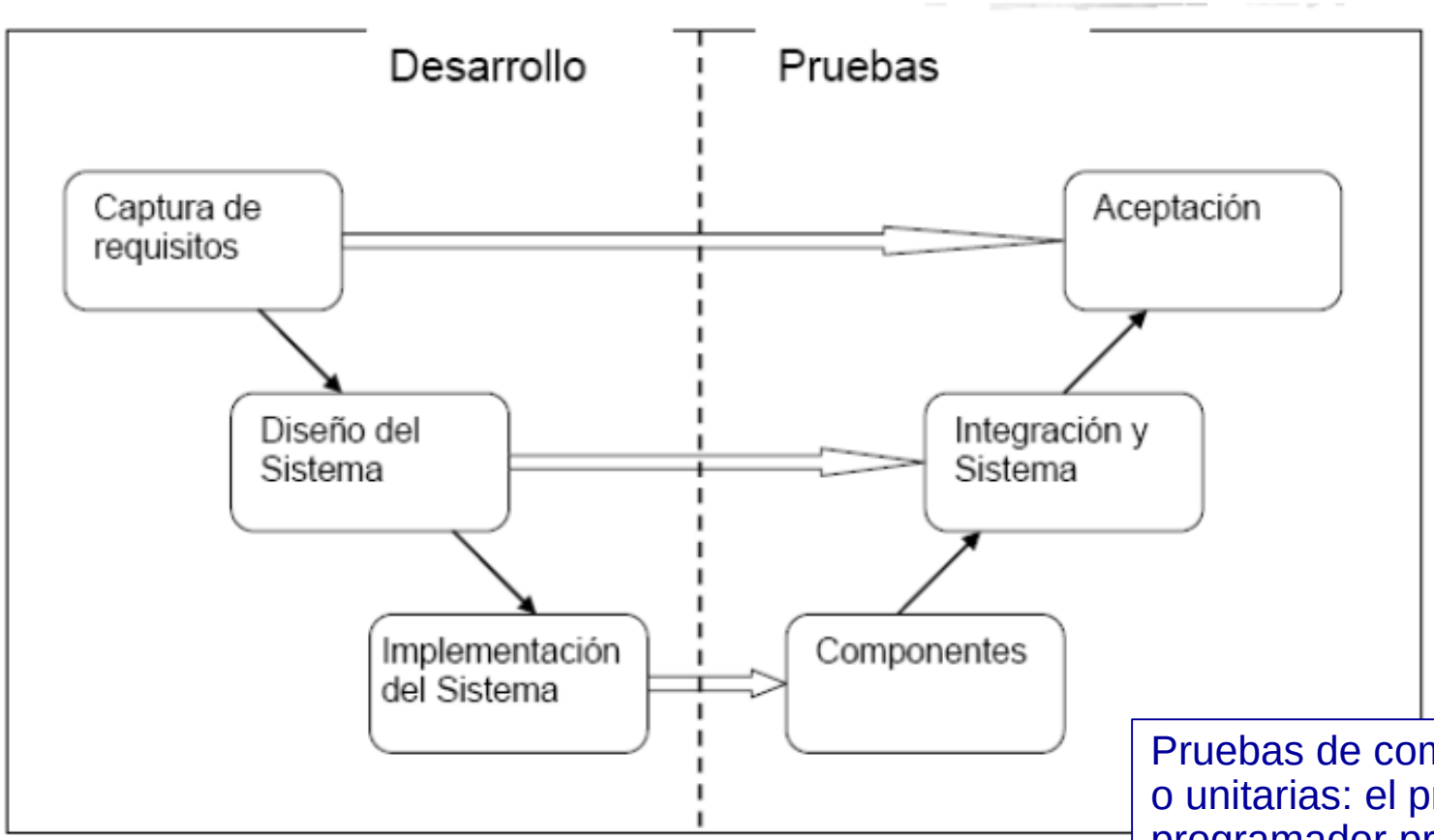


Caso práctico: continuación

- ¿Qué sucedería si el código tiene que sufrir un cambio para comprobar el stock del producto en un número variable de almacenes?
- Actividades:
  - Rehacer el código (introducir un bucle).
  - Se debería cambiar la especificación del software ¿verdad?
  - ¿Ha cambiado la interfaz del método?
  - Diseñar nuevos casos de prueba.

¿Diferencias entre pruebas de componentes o unitarias vs pruebas funcionales y aceptación?

Procesos de desarrollo y procesos de pruebas



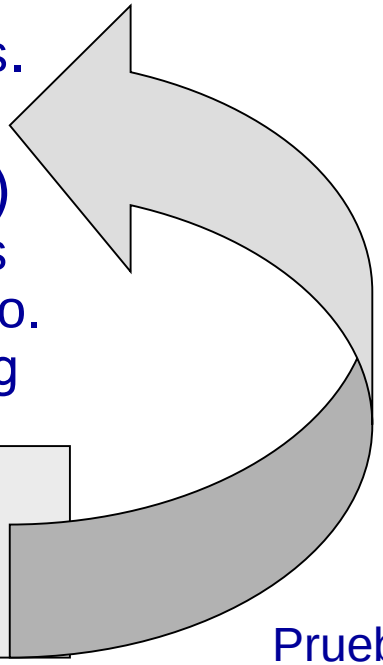
Pruebas de componentes o unitarias: el propio programador prueba y depura (habitualmente).

Diseño de pruebas: técnicas y criterios

- No es factible ejercitar el software de todas las formas posibles.
- Por tanto, se debe seleccionar (diseñar) un conjunto razonable de casos.
- Una **técnica de prueba o criterio de prueba** es un procedimiento por el que se decide qué casos de prueba se deben ejecutar sobre el software.
  - Maximizando la probabilidad de encontrar fallos.
  - Cumpliendo límites de tiempo y coste.
- Pueden diseñarse casos de prueba a partir de diferentes fuentes:
  - Especificación (requisitos del sistema, casos de uso, modelos estructurales, dinámicos u otros artefactos).
  - Documentos de diseño.
  - Código.
  - Otros (p.ej. usuarios, versiones anteriores…).

Proceso simplificado

1. Desarrollo.
2. Diseño de pruebas
    1. Aplicando diferentes técnicas y criterios.
3. Ejecución de pruebas
4. Comunicación de fallos (“reporting”)
    1. El ingeniero de pruebas comunicará los fallos detectados al equipo de desarrollo.
    2. Se pueden utilizar herramientas de “bug tracking”.
5. Detección y corrección de defectos
    1. El equipo de desarrollo detectará y corregirá los defectos.



Pruebas de regresión

Conclusiones

- Para obtener software de **calidad**, es imprescindible realizar **pruebas**.
- Las pruebas consisten en **ejecutar** el software para evaluar los resultados obtenidos.
- Si los resultados no son los esperados, habremos detectado un **fallo** (en pruebas funcionales habrá que **reportarlo**).
- Se deben realizar actividades relacionadas con los procesos de pruebas desde el inicio del proyecto (no sólo tras codificar).